

# Autonomous Emergency Landing for Fixed-Wing Aircraft with Energy Constrained Closed-Loop Prediction

Samuel J. Deal\*, Hayden L. Nichols†  
*Georgia Institute of Technology, Atlanta, GA, 30332, USA*

Anirban Mazumdar‡  
*Georgia Institute of Technology, Atlanta, GA, 30332, USA*  
*Sandia National Laboratories, Albuquerque, NM, 87123, USA*

This paper presents a new approach for autonomous motion planning for aircraft suffering from a loss of thrust emergency. Specifically, we show how modifications to the Closed Loop Rapidly exploring Random Trees (CL-RRT) framework combined with controlled energy dissipation can enable rapid and effective kinodynamic motion planning. This CL-RRT Glide algorithm uses closed loop prediction not only for node connections, but also to estimate the remaining energy and prune infeasible paths. This greatly speeds up the search process, which is essential for emergency situations. In addition, we improve the ability of the gliding aircraft to reach a goal position and energy state. We do so by creating a Dissipative Total Energy Control Scheme (TECS). Dissipative TECS enables the glider to lose excess altitude in order to reach a desired energy level. Simulation results illustrate how the proposed methods enable faster motion planning. We also integrate the system into a small UAV system and experimentally demonstrate autonomous glide planning and execution during a motor-failure event. This type of algorithm can primarily benefit unmanned aircraft, but can also serve to assist pilots in stressful, emergency situations.

## Nomenclature

$\phi$	= Euler attitude roll angle of aircraft.
$\theta$	= Euler attitude pitch angle of aircraft.
$\psi$	= Euler attitude yaw angle of aircraft.
$\rho$	= Atmospheric density.
$g$	= Acceleration of gravity.

---

\*Graduate Student, Woodruff School of Mechanical Engineering; sdeal3@gatech.edu.

†Graduate Student, Woodruff School of Mechanical Engineering; haynichols@gmail.com

‡Assistant Professor, Woodruff School of Mechanical Engineering, Faculty Joint Affiliate, Pathfinder Technologies, AIAA Member; anirban.mazumdar@me.gatech.edu.

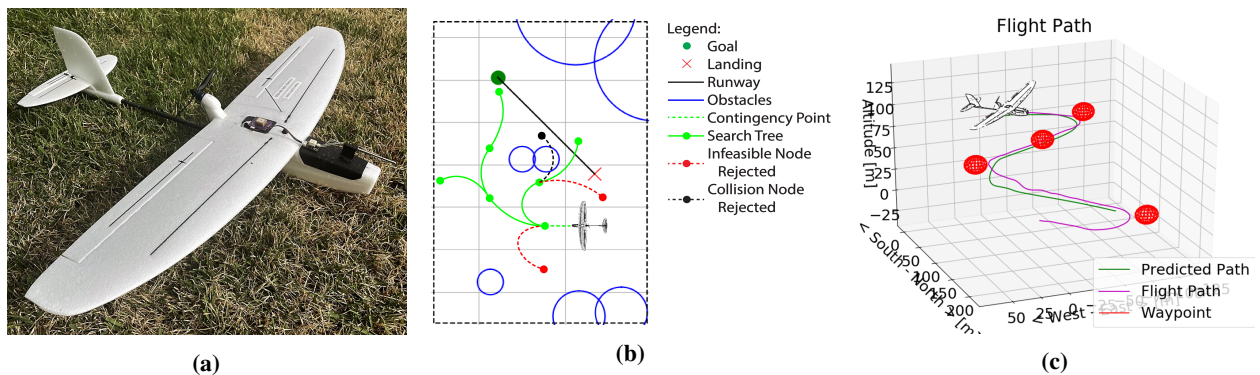
$V$  = Airspeed magnitude.  
 $V_{dem}$  = Demanded airspeed.  
 $Q$  = Dynamic pressure.  
 $L$  = Lift force.  
 $D$  = Drag force.  
 $Y$  = Side force.  
 $\ell$  = Roll moment.  
 $m$  = Pitch moment.  
 $n$  = Yaw moment.  
 $S$  = Planform area.  
 $C_L$  = Lift force coefficient.  
 $C_D$  = Drag force coefficient.  
 $C_Y$  = Side force coefficient.  
 $C_\ell$  = Roll moment coefficient.  
 $C_m$  = Pitch moment coefficient.  
 $C_n$  = Yaw moment coefficient.  
 $b$  = Wingspan.  
 $c_{mac}$  = Mean aerodynamic chord.  
 $\alpha$  = Angle of attack.  
 $\beta$  = Angle of sideslip.  
 $\phi_{dem}$  = Demanded roll angle.  
 $\theta_{dem}$  = Demanded pitch angle.  
 $B$  = Control damping.  
 $T$  = Control period.  
 $\eta$  =  $L_1$  point angle.  
 $\gamma$  = Flight path angle.  
 $a_s$  = Lateral acceleration.  
 $h$  = Altitude.  
 $h_{dem}$  = Altitude demand.  
 $\epsilon$  = Energy balance weighting.  
 $K$  = Control gain.  
 $K_d$  = Control damping gain.

- $K_i$  = Control integral gain.
- $\delta_{t_{ff}}$  = Feedforward throttle component.
- $\tau$  = TECS time constant.
- $A$  = Cross sectional area.
- $K_g$  = Glide dissipation gain.

## I. Introduction

Sudden loss of thrust can occur in aircraft due to engine damage, loss of fuel, or problems with a range of propulsion system components. This event forces the aircraft to glide, dramatically reduces aircraft range, and often necessitates rapid decision-making in order to safely land. For example, the US Airways Flight 1549 bird-strike incident started with a bird-strike and ended with an emergency landing only 210s later. Post-accident studies showed that the 35s needed to identify the failure was sufficient to prevent a return to the departure runway [1]. Recent works have shown that Flight 1549 may have only had 16s to act before the best landing site was out of range [2]. Safe recovery of a gliding aircraft requires 1) failure diagnosis, 2) landing site identification, and 3) trajectory planning. Substantial past work have addressed diagnosing aircraft failures [3, 4] and identifying best landing sites from a set of runways [2, 5].

This paper focuses on the third challenge: trajectory planning for an aircraft after failure has occurred. This paper presents a novel approach to rapidly determining a motion plan that enables a safe landing while ensuring path feasibility and avoiding obstacles. Human pilots have demonstrated the ability to quickly plan and execute safe landings with loss of thrust, but this remains a highly challenging task. Autonomous aircraft cannot rely on an expert pilot to handle such situations. There is a need for motion planning algorithms that can handle loss of thrust situations and enable safe landings under such challenging conditions. Such algorithms could improve the safety of manned, remotely piloted, and autonomous aircraft.



**Fig. 1** The experimental platform, the ZOHDrift small UAV, is shown in (a). The RRT motion planner is illustrated in (b). The flight path of a successfully planned glide landing is shown in (c).

While there has been substantial past research into motion planning for aircraft, the examination of rapid motion

planning for gliding flight remains less explored [6] [7] [8] [9] [10]. The application of path planning for a gliding aircraft presents some unique constraints for path planners. First, range and flight time are severely reduced, creating a need for methods that generate motion plans very quickly. Second, aircraft without thrust are restricted to relying on gravitational potential energy in order to sustain flight. While potential and kinetic energy can be traded for each other there are additional constraints on the aircraft velocity. High speeds result in faster energy dissipation, but the lift-to-drag ratio usually varies with velocity. The gliding speed must also lie between certain bounds to prevent stall (too slow) or structural damage (too fast). Finally, damaged aircraft must avoid certain areas to ensure the safety of those on the ground [11]. Individuals and property on the ground can be harmed by loss of control due to the emergency, falling debris, intentional fuel-dumps. This means that the desired planner must be able to consider obstacles and ensure they are avoided.

These unique challenges limit the types of planning algorithms that can be used for a loss of power situation. Planning speed, avoiding obstacles, and respecting the vehicle constraints are more critical than achieving optimal path cost. Determining the ability of a gliding vehicle to reach the goal is a necessary part of planning feasible glide trajectories. The safety of the final flight path depends on an accurate prediction of the final altitude of the flight path. If the path plan results in a final altitude that is too high, then the aircraft will be unable to execute a landing; if the altitude is too low, then it will crash. While many motion planners use simplified dynamics or heuristics to predict the trajectory, these approximations are not suitable for the many real-world cases. This is because prediction errors can cause the aircraft to crash, land in the wrong place, or fly in a manner that risks the safety of those on the ground. Therefore, we believe motion planning with forward prediction is most appropriate. Such planners like Closed Loop Rapidly Exploring Random Trees (CL-RRT) create motion plans by propagating control inputs through an accurate dynamic model [12]. This enables predictions of the aircraft states and overall energy. Such planners generate feasible trajectories relatively quickly, but optimization takes substantially longer. For loss of power situations, we are satisfied with any suboptimal, feasible trajectories that enable safe landing. The experiments presented in this paper produced a feasible trajectory for a small UAV system and a low performance processor. The feasible trajectory is generated in an average time of around 3.5 seconds using the onboard processor. This cost the UAV on average 5 meters of the initial starting altitude of 60 meters above the goal, but the UAV was still able to consistently reach the goal a horizontal distance of 215m away during multiple tests of the algorithm.

This work presents and experimentally validates a new planning methodology that is tailored specifically to sudden loss of power and resulting gliding flight, highlighted in Fig. 1c. Our proposed algorithm has the following features:

- 1) If a feasible path exists for the closed loop system, the algorithm is probabilistically complete; that is, the feasible path will be found in finite iterations.
- 2) Only a feasible path to the landing site is returned.
- 3) The return path is safe from collisions with known obstacles.

4) The vehicle reaches the goal at the correct energy state such that the standard error  $|\frac{V^2}{V_{dem}^2} - 1| + |\frac{h}{h_{dem}} - 1| < \eta$ , where  $\eta = 0.1$  is the error threshold.

5) The planning algorithm generates motion plans within seconds.

The new contributions this paper makes to the existing research are 1) the addition of a path cost heuristic based on forward simulation, 2) the design and use of a novel energy control scheme, Dissipative TECS, for rapid dissipative trajectories, and 3) the experimental validation of the algorithm.

The following presents a variation on the closed loop random sampling method, adapted for aircraft suffering from loss of thrust, highlighted by the illustration in Figure 10b. The planner adds heuristic-based pruning based on simulating the flight dynamics to the goal from each node sample. If the total path cost exceeds a threshold, the node is considered infeasible and rejected. A glide controller allows the dissipation of excess energy on the way to the goal. Finally, the algorithm is implemented on an experimental autonomous aircraft shown in Figure 10a to validate its performance during a controlled loss of thrust scenario.

## II. Background

### A. General Motion Planning

The path planning problem can be broken down into three generalizable sub-areas: the completeness and optimality of the search algorithm, the efficiency and feasibility of the node connections, and the cost estimation of the system. Many popular planning algorithms fall into two categories, either graph search algorithms like A\* and Theta\*, or random sample algorithms like probabilistic road maps (PRMs) and Rapidly-exploring Random Trees (RRT) [13] [14] [15] [16]. RRTs were developed as an alternative for quickly searching large sample spaces in a non discrete manner [16]. While probabilistically complete, the basic RRT implementation makes no guarantee of optimality. There have been now common RRT variations developed that provide asymptotic optimality. RRT\* connects new samples to the lowest cost near node and then rewires other near nodes to create a lower cost path [15]. RRT# has been shown to do an even better job at finding an optimal path than RRT\* by processing the optimization step as a separate queue of nodes most likely to be in the optimal path [17]. However, modifications are needed to account for dynamic system constraints. Path planners with nonholonomic constrained trajectories, motion primitives, and closed loop forward prediction have been proven to give feasible and efficient node connections [10] [18].

There are a few cases where path planning has been implemented on fixed-wing aircraft. One common approach is to use a motion primitive based architecture for generating dynamically feasible paths that are pre-computed for rapid path planning [19]. The work in [20] showed an application of motion primitives that connect nodes of an RRT path planner to navigate an agile fixed wing aircraft to maneuver around tight spaces and obstacles towards the goal. An RRT path planner was also used in [21] to plan for a fixed wing aircraft navigating through an urban space. Primitives have

also been suggested for path planning in situations where an aircraft becomes damaged or disabled, such as in [22].

For complex, nonlinear, multi-DoF systems, such as aircraft, forward simulation of the vehicle dynamics is a promising approach for motion planning. This enables prediction of the closed loop response to transitory periods and disturbances which may not be possible to compute off-line [18] [23] [24]. Such methods can also incorporate uncertainty. The works of [25] and [26] demonstrate an application of a chance constrained RRT planner, which builds on CL-RRT by propagating the mean and covariance of the state distributions. The ability to accurately predict the vehicle state and trajectory make closed-loop prediction an effective tool. Past works have utilized CL-RRT on quadcopters [27] and fixed-wing aircraft [28], but have not examined loss of thrust scenarios. In a loss-of-thrust event, motion planners can use closed-loop prediction to ensure a feasible and safe landing.

## **B. Handling Loss of Thrust**

We have identified random sampling plus closed-loop prediction as an effective planning method for loss of thrust. However, several additional technical challenges must be addressed in order to ensure effective planning. First, an important piece of energy-constrained path planning is a good cost estimate method. This estimate has multiple uses when planning paths. Traditionally, a running cost estimate has been used to keep track of the cost to get from one node to another [23]. For energy constrained path planning, it can serve a different purpose. Closed-loop prediction can be used to compute the reachability of not only a node, but the goal itself [29]. At any point, this makes sure that there is still enough energy left over to glide to the goal from a sampled node during the planning process. Rich literature already exists on gliding flight. However, we have identified some areas for potential improvement. Previous algorithms have not accounted for constraints when expanding the search tree to ensure that the goal is reachable from each node and on the final path. The work in [5] determines the reachability of the goal for a loss of thrust scenario prior to planning the path, but this does not take into account scenarios where obstacles might make the goal out of range. There has been other work done for glide path planning using three-dimensional dubins paths [30]. This method provides rapid solutions to the glide path and is well-suited to obstacle-free environments. In contrast, the method proposed in this paper is well-aligned with scenarios that feature obstacles or complex vehicle dynamics. Note that the analytical methods in [30] could be combined with an RRT as an initial check in case such a simple solution is possible. Finally, a different approach to path planning is to create an initial route that allows for safe landing sites in an emergency situation along the way [31].

Altitude management is another important challenge for autonomous gliding aircraft. A gliding aircraft that has no thrust loses its ability to sustain altitude under its own power, and it must gradually descend to maintain its airspeed. There is also the need to manage the aircraft altitude and dissipate excess energy on the descent to reach a target or landing site at a specific lower altitude. Previous work has examined using neural networks to determine optimal glide trajectories, but this only minimizes altitude loss rather than managing excess altitude that an aircraft might have [6] [32].

There have also been attempts at designing different trajectories that are intentionally sub optimal, so the aircraft loiters to a lower altitude [33] [9]. These paths make planning more difficult since they steer the aircraft away from the desired path, create more chances for collisions, and do not allow for feedback control of altitude to account for disturbances. Spoilers, slipping flight, and other control surfaces can be used for feedback control of altitude loss rate, but not all aircraft, especially low-cost drones, have these control inputs and these add additional complexity to the vehicle.

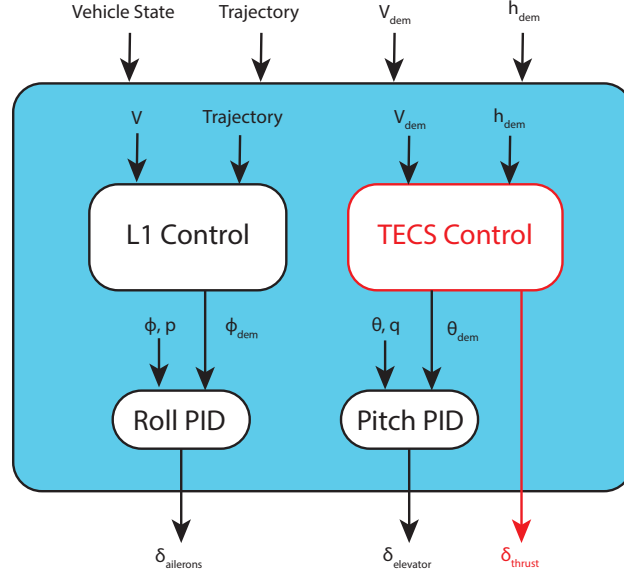
In summary, previous work in path planning has yet to address the unique additional challenges associated with sudden loss of thrust. Existing feasibility checks do not ensure that all nodes can reach the goal. In addition, feedback control for managing altitude and energy for a glider, and dissipating it in a safe manner using only elevator control input, have yet to be proposed. This work aims to address these limitations by modifying a random sample path planner to prune nodes that are out of range of the goal, adding an energy dissipation control strategy, and implementing this algorithm in real time on an experimental UAV platform. The subjects of failure detection, landing site selection, wind effects, and obstacle perception, are relevant to this work but are outside the current scope of this project. These could be modeled and incorporated in the closed-loop prediction of the path planner, which would be the subject of future research.

### **III. Fixed Wing Aircraft Dynamics and Control**

This work focuses on enabling safe-landings of fixed-wing aircraft suffering from loss of power. We assume full 6-DoF aircraft dynamics and neglect flexible modes. The aircraft is capable of regulating roll, pitch, and yaw, but lacks thrust. We use standard aircraft terminology to describe the aircraft states and controls.

Our planning architecture focuses on determining the trajectory that ensures the aircraft XYZ trajectory and speed are appropriate for a safe landing. This means that the aircraft maintains controlled flight, does not collide with terrain, avoids no-go areas, and lands at a safe speed. Other dynamic constraints such as structural or acceleration limits can be easily incorporated.

There are two strategies that are used to independently control the lateral and longitudinal aircraft dynamics. The lateral control primarily determines how the aircraft turns in the horizontal plane. This is useful when following simple two dimensional trajectories, like a straight line between way-points or a circular turn. The longitudinal control determines how the aircraft manages its kinetic and potential energy in the vertical plane. These two controllers limit the turning, climbing, and descent rates of the aircraft. The path planner accounts for these controllers during the forward simulations. As Fig. 2 illustrates, an  $L_1$  controller [34] is used for lateral control while the Total Energy Control System (TECS) is used for longitudinal control. These two controllers combine to generate roll and pitch set-points that are then passed to respective attitude PID controllers that determine the resulting control surface deflections. This architecture is similar to that in [27].



**Fig. 2 Overall waypoint controller diagram. The  $L_1$  controller steers the aircraft on a path heading towards the waypoint, while the TECS controller balances the demanded speed and climb rate. The items highlighted in red show what is absent when the TECS Control is switched to the Dissipative TECS.**

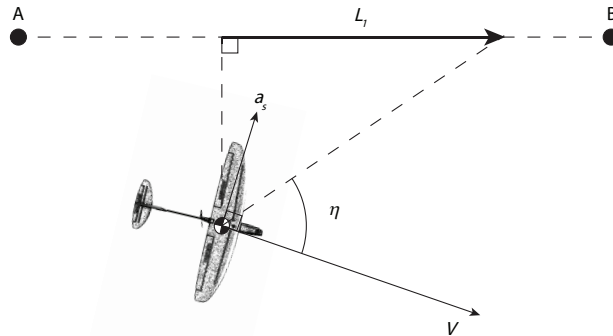
### A. $L_1$ Controller

For the lateral control, an established method is chosen based on the  $L_1$  non-linear control strategy developed in [34] and utilized in the Ardupilot open source flight controller project for automatic flight modes. It has been shown to work very well with small unmanned aerial vehicles and is widely utilized in open-source software [35]. The  $L_1$  controller takes in the airspeed  $V$ , the heading relative to the  $L_1$  point  $\eta$  on the desired trajectory shown in Fig. 3, and accounts for the pitch angle of the aircraft  $\theta$ . The controller computes a desired lateral acceleration,  $a_s$ , and then maps this to a desired roll angle,  $\phi_{dem}$ . The basic  $L_1$  control equations are shown in 1 - 3. The aggressiveness of the controller is based on the length of  $L_1$  along the desired flight path ahead of the vehicle; the shorter that  $L_1$  is, the more aggressively the controller pulls the vehicle towards the trajectory. The damping  $B$  and period  $T$  terms are used to tune the aggressiveness of the controller. Overall, the commanded lateral acceleration,  $a_s$ , is directly proportional to the vehicle velocity, so the controller adjusts to account for the airspeed of the vehicle. This helps it to reject disturbances from wind and follow the trajectory more closely. The  $L_1$  point also gives the controller an element of anticipation of the path, enabling it to perform better than pure linear proportional derivative controllers when following a curved path.

$$L_1 = \frac{1}{\pi} BTV \quad (1)$$

$$a_s = \frac{4B^2V^2 \sin(\eta)}{L_1} \quad (2)$$

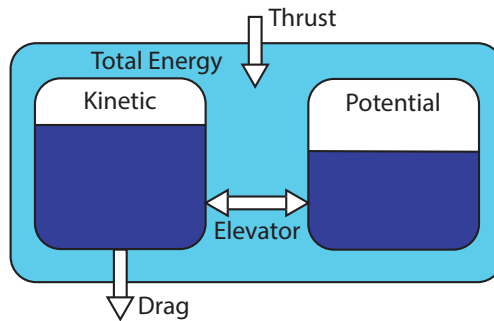
$$\phi_{dem} = \cos(\theta) \arctan(a_s/g) \quad (3)$$



**Fig. 3** Diagram illustrates the L1 controller. Resultant lateral acceleration  $a_s$  drives the aircraft towards the desired path. The control action is overall proportional to the sine of the path angle  $\eta$ .

### B. TECS Controller

This work uses a type of longitudinal control strategy known as the Total Energy Control System (TECS), which attempts to manipulate the energies of the vehicle shown in Fig. 4. The principle for this type of control is presented in [36]. This controller has also been widely used in open-source software [35]. The TECS takes in the aircraft's velocity  $V$  and altitude  $h$  state, as well as the demanded velocity  $V_{dem}$  and altitude  $h_{dem}$ , and outputs the demanded pitch angle  $\theta_{dem}$  and the thrust setpoint  $\delta_t$ .



**Fig. 4** Diagram shows the transfer of potential and kinetic energy. Thrust affects the total energy, while drag removes kinetic energy. The elevator controls the difference between the kinetic and potential energy.

With the standard TECS controller, the aircraft thrust is used to add energy to the overall system, and the pitch control is used to trade energy between kinetic and potential. Generally, thrust is used to add energy to the overall system. This means that the control state for the thrust PID-FF controller in Equation 7, is the error between the demanded and

actual aircraft energy.

$$\mathbf{te} = \frac{1}{2}V^2 + gh \quad (4)$$

$$\mathbf{te}_{dem} = \frac{1}{2}V_{dem}^2 + gh_{dem} \quad (5)$$

$$\mathbf{te}_{err} = \mathbf{te}_{dem} - \mathbf{te} \quad (6)$$

$$\delta_t = K(\mathbf{te}_{err} + K_d \dot{\mathbf{te}}_{err} + K_i \int \mathbf{te}_{err} dt) + \delta_{t_{ff}} \quad (7)$$

The pitch controller uses a weighted difference to compute the energy "balance",  $\mathbf{eb}$ , or the difference between the potential and kinetic energies. The weighting is given by  $\epsilon \in [0, 2]$  in Equation 9, which determines what errors the pitch control favors. When  $\epsilon = 1$ , pitch control will simultaneously control speed and height. For the case where  $\epsilon > 1$ , speed control will be more accurate while height errors will be greater, which tends to work better for lower speed, high lift, and high drag aircraft. Finally when  $\epsilon < 1$ , height control will be more accurate while speed errors will be greater, which tends to work better for higher speed, lower lift, and low drag aircraft. In the case of this work, thrust is neglected. This means that if thrust is lost, the desired energy balance cannot be maintained for  $\epsilon < 2$ , and the elevator could cause a stall by trying to maintain altitude.

$$\mathbf{eb}_{dem} = (2 - \epsilon)gh_{dem} - \epsilon \frac{1}{2}V_{dem}^2 \quad (8)$$

$$\mathbf{eb}_{err} = \mathbf{eb}_{dem} - [(2 - \epsilon)gh - \epsilon \frac{1}{2}V^2] \quad (9)$$

$$\theta_{dem} = \frac{1}{V\tau g}(\mathbf{eb}_{err} + K_d \dot{\mathbf{eb}}_{err} + K_i \int \mathbf{eb}_{err} dt + \tau \mathbf{eb}_{dem}) \quad (10)$$

### C. Dissipative TECS

The TECS framework requires modifications for use with a vehicle that lacks thrust. We propose a Dissipative TECS, which enables the gliding aircraft to reach lower altitudes quickly. First, the energy balance factor  $\epsilon = 2$  which eliminates the altitude component from the energy balance equation in 8 and 9. This completely eliminates altitude control and only uses the elevator to control speed. With just this modification, the Dissipative TECS causes the gliding

aircraft to climb in order to absorb excess airspeed, and descend in order to maintain the demanded airspeed. During the majority of flight, the velocity demand should be set to the velocity that maximizes lift to drag ratio. This maximizes the aircraft range.

Regulating altitude is challenging because altitude and speed are dependent, and the lack of thrust means that altitude can not be fully controlled. Changing the pitch angle of the aircraft changes its airspeed and climb rate. Increasing the steepness of the glide angle increases the airspeed until the lift, drag, and gravitational forces are balanced. Decreasing the steepness the glide angle to the minimum glide angle decreases the airspeed, but attempting to glide flatter than the minimum glide angle causes a stall since the forward component of the gravitational force is not enough to overcome drag and maintain the necessary airspeed and lift force.

In order to successfully land a gliding aircraft, all excess altitude must be eliminated by the time it reaches the desired landing site and the aircraft must be at a safe landing speed. Without a dissipative controller that simple drones can use, the planning algorithm would have to create a path of the exact length necessary to dissipate the correct amount of altitude. This would mean that a planning algorithm would have to spend a lot of time searching paths that would reach the goal except for being a little too high. Having a feedback altitude controller such as the one proposed below provides a convenient way for a closed-loop planning algorithm to reach the final goal altitude.

Equation 11 shows the drag force equation which determines the dissipative force as a result of motion through a fluid. Combining this with the power Equation in 12 creates Equation 13, which determines the rate of energy loss due to drag. The key observation from this equation is that energy is lost at a rate proportional to the cube of the vehicle airspeed  $V$ . Therefore, increasing the aircraft airspeed for a period of time is an extremely effective method of dissipating excess energy in order to achieve a desired speed at a lower altitude.

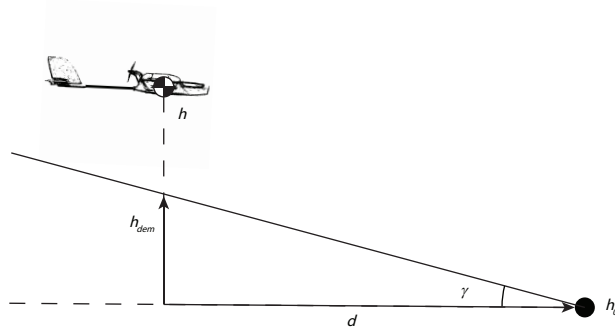
$$D = \frac{1}{2}\rho AV^2 C_d \quad (11)$$

$$P = \vec{F} \cdot \vec{v} \quad (12)$$

$$P_{loss} = \frac{1}{2}\rho AV^3 C_d \quad (13)$$

We create a Dissipative TECS by adding a predictive component to the TECS to determine the demanded altitude for the glide trajectory. Fig. 5 and Equation 14 show how the altitude demand is estimated. The excess altitude should be calculated relative to the ideal glide trajectory that intercepts the goal. The Dissipative TECS uses the glider's pitch control to increase the airspeed proportional to the excess altitude until the total energy of the glider equals the total energy demanded. The desired control behavior pitches the aircraft down until it approaches the desired glide path.

After the excess energy is dissipated, the glider pitches back up and finishes gliding towards the next waypoint while it returns to the desired glide airspeed



**Fig. 5** Diagram shows the control variables of the Dissipative TECS, which increases the drag with a higher airspeed until the height  $h$  of the aircraft equals the  $h_{dem}$  on the desired glide path angle  $\gamma$ .

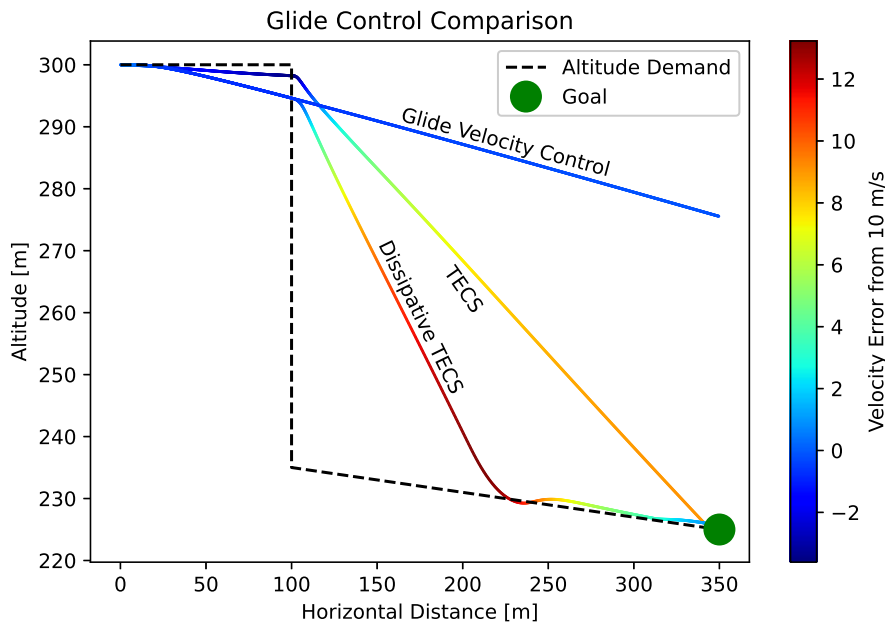
$$h_{dem} = d \tan(\gamma) + h_0 \quad (14)$$

This control behavior is achieved by modifying the TECS pitch control demand according to Equation 15 and 16. The normal behavior of the unmodified TECS for a glider is to completely sacrifice altitude control to maintain the demanded glide airspeed by setting  $\epsilon = 2$ , which results in the upper condition in Equation 15. This maximizes the glide range when the aircraft energy is less than or equal to the energy demand,  $te \leq te_{dem}$ . However, to dissipate excess altitude potential energy, the gravitational potential energy must be taken into account. This is accomplished in the second condition of Equation 15 by making the energy balance proportional to the potential energy difference of the glider if the total energy is greater than the energy demand,  $te > te_{dem}$ . The glide gain  $K_g$  has the effect of emphasizing the altitude error in the pitch demand control equation. The result is that excess altitude causes a rapid pitch down and subsequent increase in velocity and when the total energy of the vehicle approaches the total energy demand, the glider switches to the glide velocity control. The reason for this dual-mode action of the Dissipative TECS is that once the glider goes below the demanded energy it has no way of returning to the higher energy state. Thus it is very important that once the energy balance approaches the total energy demand, then it needs to return to a normal glide to sustain the correct amount of energy to reach the goal altitude through the remainder of the flight. One additional note is that Equation 10 is identical to Equation 16 except that the last term in the former equation,  $\dot{e}b_{dem}$ , is zero for a gliding vehicle. This is because  $V_{dem}$  will be at a constant best glide velocity for a gliding vehicle.

$$eb_{err}^* = \begin{cases} (-V_{dem}^2 + V^2), & te \leq te_{dem} \\ -K_g(gh_{dem} - gh), & te > te_{dem} \end{cases} \quad (15)$$

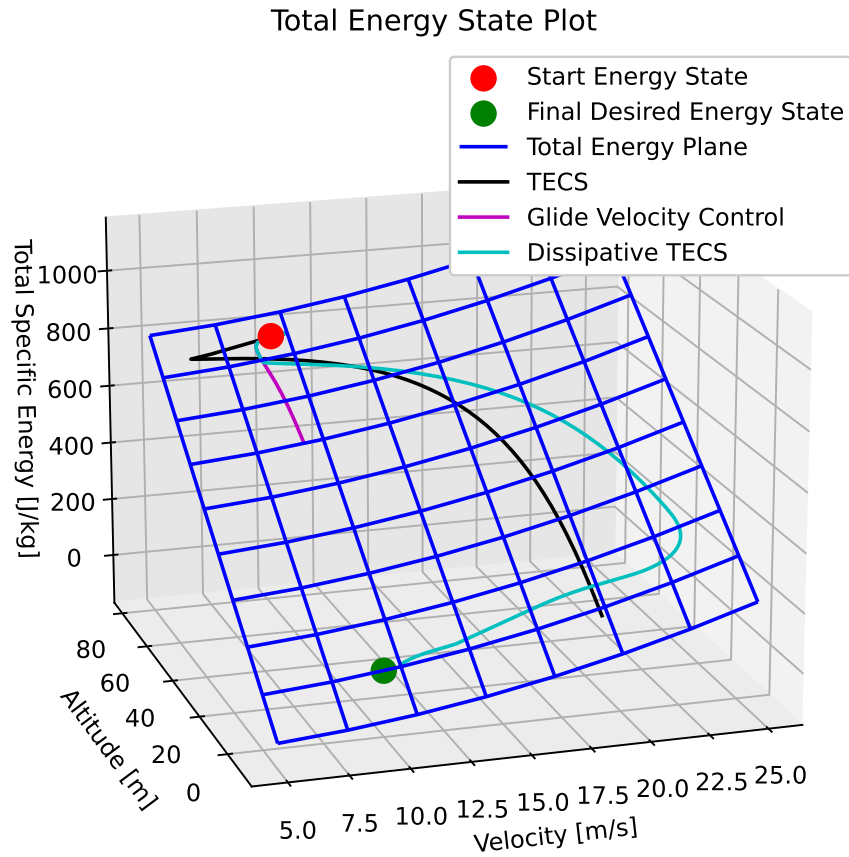
$$\theta_{dem} = \frac{1}{V\tau g} (eb_{err}^* + K_d eb_{err}^* + K_i \int eb_{err}^* dt) \quad (16)$$

A simulation comparison between the three different behaviors of the longitudinal controllers is shown in Fig. 6. A simple course was set up with two segments separated by a step change in altitude. We also want to reach the final altitude at a desired velocity of 10 m/s. The first control system tested is the unmodified TECS with  $\epsilon = 0.3$ , which is used for normal flights with thrust, to see how it behaves during glide. The second control system is the same TECS controller but with the  $\epsilon = 2$  which results in a glide velocity controller. This is how the TECS controller is normally modified when there is no thrust to maintain a constant glide airspeed. The third control is the Dissipative TECS for altitude loss control. This uses the pitch control to increase the glide velocity for a brief period to rapidly dissipate excess energy when approaching the goal altitude. The aircraft velocity relative to the desired glide velocity is shown by the color gradient.



**Fig. 6 Plot of glide control performance compares the behaviors of the unmodified TECS and the Dissipative TECS. The unmodified TECS control flies a straight path with excess kinetic energy. The Dissipative TECS initially loses the excess altitude to reach the final glide path to the goal.**

The TECS control with  $\epsilon = 0.3$  does not behave in a desirable manner in the absence of thrust, as revealed by the simulation altitude plot Fig. 6. Flying during the initial phase where the altitude demand was constant, the aircraft was nearing stall as the pitch was trying to correct for the altitude error. In addition, near the goal altitude, the glide speed increased dramatically. The Glide Velocity Control maintained the desired glide velocity, but ignored the altitude command. Finally, the Dissipative TECS controller showed that it effectively dissipated most of the excess energy



**Fig. 7** Plot of the energy state trajectories for each glide control path from Figure 6. The blue plane shows the surface of all possible total specific energy states vs. the glider velocity and altitude.

after the step change in altitude demand without having to fly a long path. The controller rapidly pitched down which increased the flight velocity at first. Then the controller quickly returned to the desired glide speed and trajectory by increasing the pitch angle again when the total glider energy approached the estimated energy demand. The energy state trajectories from each of these controllers is shown by the plot in Fig. 7, which illustrates how closely each controller can get to the final goal state. It becomes clear how the TECS and Glide velocity controller end their paths too fast and too high respectively. The Dissipative TECS is able to reach the goal altitude, albeit with a small error in the velocity of the glider of 0.84 m/s for this case. Although this velocity error could be problematic for a landing approach, the dissipative control is best suited for the penultimate stage of the gliding flight. For this phase of the flight, an aircraft needs to reach a waypoint at a specified altitude in line with the runway in order to set up for a landing, leaving plenty of time for a landing controller to make sure the aircraft is approaching at a safe landing speed.

#### IV. Planning Algorithm

This section outlines the proposed path planning algorithm for the thrust loss scenario of a fixed wing aircraft. The proposed algorithm, CL-RRT Glide, is a modified version of the CL-RRT algorithm proposed in [18]. It utilizes the same principle of sampling points in the navigation control space (the set of all points reachable by the vehicle's navigation controller) and forward simulating the vehicle dynamics to determine the feasible vehicle path between nodes. In this work, the navigation control space is the XYZ inertial position states of the aircraft. Since the goal of this algorithm is to land the aircraft safely on a predetermined runway, the final stage of the flight path will already be known. The algorithm will determine how to navigate the aircraft to the location and altitude to set up for the final landing approach. The planner samples and connects XYZ position states. The full 6-DOF vehicle dynamics are simulated, but only the XYZ position states are used for sampling and node connections. In addition, in this work, we often only sample in the XY plane and allow Z to vary. We constrain Z when checking the feasibility of the goal. The algorithm will naturally plan paths that are not optimal because of the random sampling, which will cause some loitering behavior that allows the aircraft to lose excess energy. It will also take advantage of the Dissipative TECS discussed in the previous section to reach the correct final altitude for an approach to the runway. This will reduce the amount of sampling that the planning algorithm will have to do in order to reach a low enough altitude. It should also be noted that the reason the Dissipative TECS is not used for reaching every intermediate node is that there is no specified altitude goal for these nodes. Therefore, it is proposed that the glider should fly as efficiently as possible during the intermediate waypoints because this could allow future improvements to the planner to account for unforeseen situations like wind or dynamic obstacles.

Forward simulation can also be used to generate a high-fidelity heuristic that can be leveraged for more efficient planning. This allows the planning algorithm to account for all of the modeled effects of the dynamics when planning paths, such as changes to glide path angle due to banking, to ensure that a trajectory can reach the goal. Specifically,

an upper bound can be generated for the cost to the goal. Leaves of a branch whose cost plus the cost heuristic to the goal exceeds the threshold are trimmed from the tree. The cost restriction ensures that the goal is feasible: given enough samples of potential paths, if there exists a feasible path to the goal, the path planning algorithm will find it. When sufficient constraints are placed on the planner, it is possible to rapidly generate efficient paths with better consistency than a pure random search algorithm. While the algorithm does not find an optimal path, it is faster and more computationally simple than an algorithm that progressively optimizes. For emergency type scenarios, we believe that feasible planners are more appropriate than optimal ones. Additionally, the planner can find a feasible solution for the dynamics so long as one exists, which we refer to being probabilistically feasible. Even though it is not an optimal planner it will still successfully find paths even in a demanding scenario where the minimum distance to the goal is right at the limit of the glide range, although these solutions will be less inherently robust to disturbances like wind.

### A. Problem Formulation

Given the sets  $\mathbf{X} \subseteq \mathbb{R}^{n_x}$ ,  $\mathbf{Y} \subseteq \mathbb{R}^{n_y}$ , and  $\mathbf{U} \subseteq \mathbb{R}^{n_u}$ , the nonlinear dynamics of the vehicle are assumed to be of the form described by Equation 17

$$\begin{aligned}\dot{\mathbf{x}}(t) &= f(\mathbf{x}(t), \mathbf{u}(t)), & \mathbf{x}(0) &= \mathbf{x}_0 \\ \mathbf{y}(t) &= h(\mathbf{x}(t), \mathbf{u}(t))\end{aligned}\tag{17}$$

where  $\mathbf{x}(t) \in \mathbf{X}$  are the vehicle states,  $\mathbf{u}(t) \in \mathbf{U}$  are the control inputs,  $\mathbf{y}(t) \in \mathbf{Y}$  are the output states,  $\mathbf{x}_0 \in \mathbf{X}$  are the initial conditions of the vehicle states, and  $f$  and  $h$  are continuously differentiable functions representing the system dynamics in time. Let  $\mathbf{X}_{obstacle} \subseteq \mathbf{X}$  be the subset of vehicle collision states,  $\mathbf{X}_{goal} \subseteq \mathbf{X}$  be the subset of vehicle goal states, and  $\mathbf{U}$  be the set of allowable control inputs. On the interval  $[0, T]$ ,  $T \in \mathbb{R}$ , let  $\mathcal{X}$ ,  $\mathcal{Y}$ , and  $\mathcal{U}$ , the sets of all bounded functions that map to the state, output and control trajectories respectively. The vehicle control is assumed to track a reference trajectory  $\mathbf{r}(t)$  given a desired output value  $\hat{\mathbf{y}} \in \mathbf{Y}$  and a current output value  $\mathbf{y} \in \mathbf{Y}$  based on a control function  $\Phi : (\hat{\mathbf{y}}, \mathbf{y}) \mapsto \mathbf{u} \in \mathbf{U}$ .

The goal of the motion planning problem given the above system in Equation 17 is to determine a reference trajectory  $\mathbf{r} \in \mathcal{Y}$ ,  $[0, T]$ ,  $T \in \mathbb{R}$  such that the state and output trajectories  $\mathbf{x} \in \mathcal{X}$  and  $\mathbf{y} \in \mathcal{Y}$  produced by the closed loop response of the control sequence  $\mathbf{u} \in \mathcal{U}$  meet the following criterion:

- 1) obey the system dynamics in Equation 17,  $\forall t \in [0, T]$
- 2) avoid collisions, that is  $\mathbf{x}(t) \notin \mathbf{X}_{obstacle} \forall t \in [0, T]$
- 3) ensure that the goal region is in reach, that is  $\forall \mathbf{x}(t), \exists \mathbf{r}, | \mathbf{x}(T) \in \mathbf{X}_{goal}, t \in [0, T]$ ,
- 4) reach the goal region,  $\mathbf{x}(T) \in \mathbf{X}_{goal}$

## B. Procedures

The following are detailed descriptions of the main procedures used in Algorithm 1:

*Sample Space:* `Sample` :  $X \mapsto \hat{y} \in Y$  function returns uniformly distributed independent random desired output vector sample  $\hat{y}$  of the state search space  $X$ . This random sample is used to produce the reference input to the controller, which for this purposes of this project is a point in the horizontal XY plane and does not include a final heading.

*Get Nearest Neighbor:* `getNearest` :  $(\hat{y}, S) \mapsto v$  function takes the result from `Sample` and finds the nearest node  $v$  in the search tree  $S$  to the random sample  $\hat{y}$  in terms of a given distance function. The distance function is a simple heuristic used for finding the nearest node and is not used for any steering or control reference. Using an accurate distance function in RRT is important because shorter paths have a lower probability of collision. A Dubins path length is chosen for this application of the algorithm to account for the turning limitations. The minimum turning radius of the vehicle is calculated from the maximum centripetal acceleration allowed, which for the example used in this paper is  $2g$ . Because the glider can't fully control altitude and the planner is really only a 2D planner with a feasibility check in the vertical third dimension, the 2D Dubins path is used as formulated in [18].

*Steering:* `Steer` :  $(y, \hat{y}) \mapsto \mathbf{r} \mid \mathbf{r} \in \mathcal{Y}$  function takes the random output space sample,  $\hat{y}$ , and the end of the nearest node reference trajectory,  $\mathbf{r}.end$ . The function returns a reference trajectory,  $\mathbf{r}$ , connecting the nearest node (returned from `getNearest`) to the random output sample. This is the input to the controller  $\Phi$  for the closed loop dynamics. In this work, the distance that the reference trajectory was allowed to extend towards the random sample was limited to a maximum distance  $d > 0$  except in the case of the goal node, such that the euclidean distance  $\|\hat{y} - y\|$  is minimized and  $\|\hat{y} - y\| \leq d$ . This distance limitation helps avoid frequent collisions during the graph exploration, but was not used when sampling the goal node to make it faster at finding a path to the goal.

*Node Propagation:* `Propagate` :  $(\mathbf{x}, \mathbf{r}) \mapsto \mathbf{x}$  function takes the last state vector from the nearest node state trajectory  $v_{nearest}.\mathbf{x}.end$  and the reference input towards the next node  $v_{new}.\mathbf{r}$  and propagates the system dynamics forward in time. The resulting state trajectory is returned.

*Collision Check:* `CollisionFree` :  $(\mathbf{x}, X_{obstacle}) \mapsto \{0, 1\}$  Boolean function returns true if every state in the newly propagated node trajectory  $v_{new}.\mathbf{x}$  is in the obstacle free space, that is  $\mathbf{x} \notin X_{obstacle} \forall \mathbf{x} \in v_{new}.\mathbf{x}$ .

*Feasibility Check:* `Feasible` :  $(\mathbf{x}, X_{goal}) \mapsto \{0, 1\}$  Boolean function returns true if the potential final goal node trajectory state  $v_{goal}.\mathbf{x}.end \in X_{goal}$ . After every new node is propagated, the vehicle dynamics are propagated to the goal, in this case the Dissipative TECS is going to be used as opposed to the regular TECS that is used for the normal nodes that are not the goal. The feasibility check is then used to be sure that the goal is in range of the new node. If the check returns true, then this means that the newest branch is potentially part of the final path to the goal and should be added to the search tree. For the application in this paper, the goal region is a point in three dimensional XYZ space, and does not include a final orientation or heading. The feasibility is checked based on the altitude, or Z dimension, so a path that goes below some threshold of the desired final altitude is considered infeasible and the new node is pruned. If

the path goes above the goal because the Dissipative TECS could not reduce the altitude enough, then the new node is still feasible and will be added to the search tree but more samples still need to be searched until enough energy is lost for the aircraft to make a successful landing.

*Return Final Path:*  $\text{Path} : (S) \mapsto \mathbf{r}$ . Given a search tree  $S$ , function returns the final path, the controller reference trajectory  $\mathbf{r}$ , once a collision free feasible path is found to the goal node. The path is the set of all the control inputs of the parents of the final node  $v_{goal}$ , which is returned by the algorithm to be executed by the actual vehicle.

### C. Algorithm Description

The main algorithm CL-RRT Glide, detailed in Algorithm 1 and illustrated in Fig. 8, begins with the current state of the vehicle, the search space, goal space, and obstacle space given. The search tree, a directed graph where each node has a single parent that represents the set of explored nodes, is initialized with the start node as the origin. The node object data structure is given by Table 1, where there is a control reference input, a state trajectory of the vehicle dynamics in response to the reference input which represents the edge to get to the node, and a reference to the parent node that precedes the new node. The start node, which necessarily does not have a control input or a parent, has a state trajectory that consists only of the initial vehicle state. The main program loop begins sampling the search space for the vehicle. The nearest node in the search tree to the random sample is selected by a chosen heuristic, in this case a minimum length Dubins path was used. The reference control input for the closed loop dynamics is determined from the random sample. The closed loop dynamics are then propagated from the parent node towards the reference input to determine the node connection, and the new node is returned.

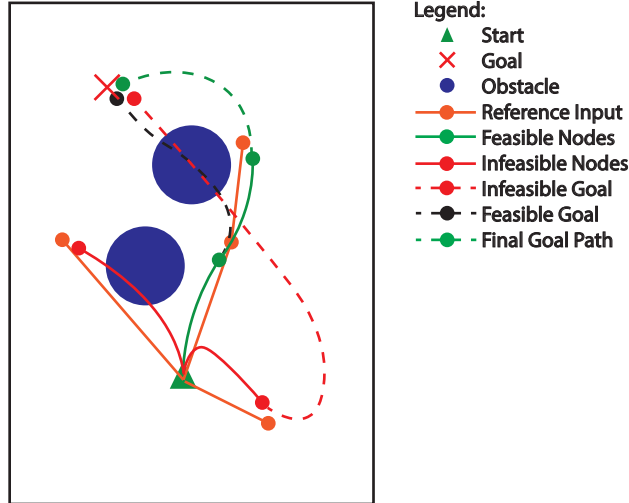
The next step checks that the state trajectory "edge" to the new node is collision free. If not, the new node is rejected and the loop starts the next iteration, but if the new node does avoid a collision, then the pruning step begins. The propagation of the closed loop dynamics towards the goal is leveraged as a high fidelity heuristic to determine the cost of the path. This cost can be associated with a range of considerations. In this work we use a cost associated with the finite energy of the gliding aircraft. If the new node cost plus the cost computed by the forward simulation towards the goal exceeds the maximum threshold, then the node is discarded and the searching loop continues. This ensures that the final path that is returned does not exceed the upper cost limit, making the algorithm probabilistically feasible. If the goal is feasible, then the new node is added to the search tree and the goal node is evaluated. If the goal node state trajectory is collision free and the trajectory end state is in the goal state space, then the main program loop returns the feasible path by linking the goal node and all its predecessors back to the start. Otherwise if the goal node state trajectory is not collision free or does not reach the goal state space, then the node is kept in the tree but more planning is needed. Therefore, the program loop begins to sample more nodes in order to complete the path. If the program loop reaches the maximum sample limit without finding a feasible path, then the end of the program returns the null set as the loop expires.

**Table 1 Search Tree Node Data Structure**

Field	Type	Description
$y$	Vector $\in \mathbb{R}^{n_y}$	Output Vector
$r$	Trajectory $\in \mathbb{R}^n$	Control Reference
$x$	Trajectory $\in X$	State Trajectory
$x.end$	Last State Vector	State Vector
$P$	Node	Parent Node

**Algorithm 1** CL-RRT Glide Algorithm

1: $y_0, x_0, y_{goal}, X, X_{goal}, X_{obstacle}$	▸ Initial algorithm inputs
2: $v_{start} \leftarrow \text{Node}()$	▸ Initialize start node
3: $v_{start}.y \leftarrow y_0$	▸ Assign start node output as initial output
4: $v_{start}.x \leftarrow \{x_0\}$	▸ Assign start node state trajectory as initial state
5: $v_{goal} \leftarrow \text{Node}()$	▸ Initialize goal node
6: $S \leftarrow \{v_{start}\}$	▸ Begin search tree with start node
7: <b>for</b> $i = 1$ to $N$ <b>do</b>	
8: $y_{rand} \leftarrow \text{Sample}(X)$	▸ Sample control space
9: $v_{nearest} \leftarrow \text{getNearest}(y_{rand}, S)$	▸ Find nearest node
10: $v_{new} \leftarrow \text{Node}()$	▸ Initialize new node
11: $v_{new}.r \leftarrow \text{Steer}(v_{nearest}.y, y_{rand})$	▸ Determine reference input
12: $v_{new}.x \leftarrow \text{Propagate}(v_{nearest}.x.end, v_{new}.r)$	▸ Propagate dynamics
13: $v_{new}.P \leftarrow v_{nearest}$	▸ Assign new node parent
14: <b>if</b> $\text{CollisionFree}(v_{new}.x, X_{obstacle})$ <b>then</b>	▸ Check if state trajectory collides
15: $v_{goal}.r \leftarrow \text{Steer}(v_{nearest}.y, y_{goal})$	▸ Determine reference input to goal node
16: $v_{goal}.x \leftarrow \text{Propagate}(v_{new}.x.end, v_{goal}.r)$	▸ Propagate dynamics to goal
17: $v_{goal}.P \leftarrow v_{new}$	▸ Assign goal node parent
18: <b>if</b> $\text{Feasible}(v_{goal}.x.end, X_{goal})$ <b>then</b>	▸ Check if goal space can be reached
19: $S \leftarrow S \cup \{v_{new}\}$	▸ Add new node to search tree
20: <b>if</b> $\text{CollisionFree}(v_{goal}.x, X_{obstacle})$	
21: <b>and</b> $v_{goal}.x.end \in X_{goal}$ <b>then</b>	▸ Check if trajectory ends in goal space
22: <b>return</b> $\text{Path}(v_{goal})$	
23: <b>end if</b>	
24: <b>end if</b>	
25: <b>end if</b>	
26: <b>end for</b>	
27: <b>return</b> $\emptyset$	▸ Return Null if algorithm fails to find path



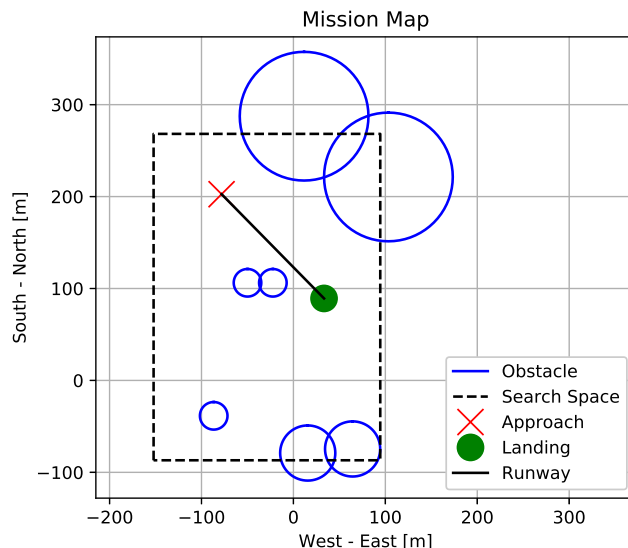
**Fig. 8 Illustration of the planning algorithm highlighting how the search tree is built. Shown is the difference between the reference control input, infeasible nodes, feasible nodes, and collision trajectories.**

#### D. Algorithm Performance

The CL-RRT Glide algorithm is compared to a basic CL-RRT algorithm to compare the performance for planning a path with a gliding vehicle. The only significant difference in the behavior of the two algorithms is the CL-RRT Glide algorithm adds the the goal sampling and cost heuristic checks for every node sample. Basic CL-RRT on the other hand only samples the goal at random intervals to see if there is a collision free path from the nearest node to the goal. It does not ensure that the goal is feasible for the vehicle, so the final path is likely to be too high or too low.

For the comparison, the ZOHD Drift fixed wing aircraft flight dynamics model with the controller described in Section III is used for the closed loop simulation as the example gliding vehicle. A sample search environment shown in Fig. 9 was created based on the actual flight field to test the algorithms' ability to find a path to the approach way-point location and altitude, which defines the position from which the aircraft can make a safe glide landing along the runway. Each algorithm was run 1,000 times with a timeout of 1000 node samples in order to evaluate how well on average the algorithm was able to perform in finding a glide path to the goal. For each algorithm run, a random starting orientation, starting location in the obstacle free search space, and altitude in the glide range of the approach, were generated to begin the path planner. From there, the path planning algorithm attempts to find a path along which the glide control would be able to navigate the aircraft to the approach position while avoiding any obstacles. If the planner fails to find a path that reaches the goal state space before the maximum number of node samples is reached, the algorithm expires rather than continue to search indefinitely.

The results of the comparison are highlighted in Table 2. With the CL-RRT Glide algorithm, every node added to the tree is forward simulated towards the goal to ensure that the vehicle can reach the goal from that node. This goal feasibility check increased the efficiency of the path planner by stopping the planner from expanding branches with

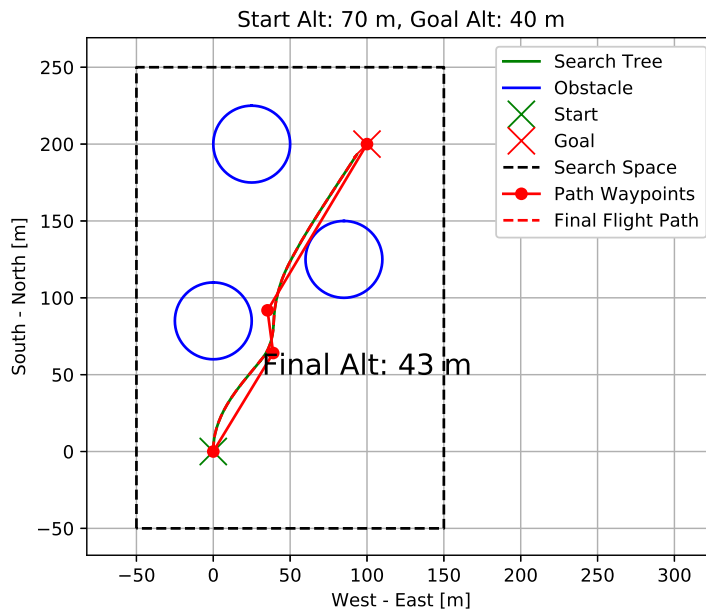


**Fig. 9** Figure shows the layout of the test mission environment. The obstacles define unsafe regions for the airplane to fly. There is an approach point that serves as the goal for the path planner from where a safe landing can proceed along the runway.

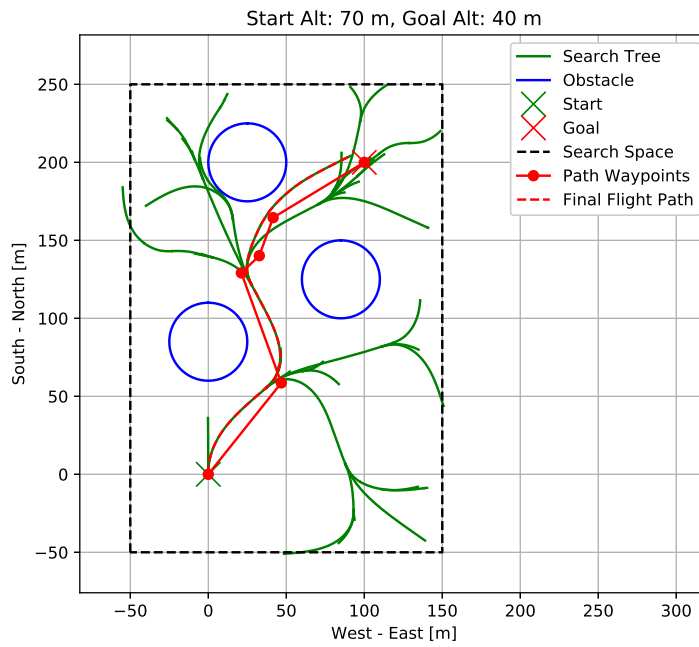
**Table 2** Algorithm Performance Comparison (1000 trials)

Planner	Success	Avg Time [s]	Avg Samples
CL-RRT	325	6.25	742
CL-RRT Glide	1000	0.0335	12.7

insufficient energy. If there is excess energy after testing the approach point, the algorithm continues sampling until the approach altitude is in the control space of the glide controller (the set of all controlled states reachable by the glide controller from a given initial condition), preventing the algorithm from returning a path that ends too high. The average number of sampled nodes were just 12.7, and the modified algorithm succeeded 100% of the time at finding a path to the goal before reaching the maximum number of node samples. The base CL-RRT algorithm only succeeded 32.5% under the same scenario because it wastes samples trying to expand nodes that cannot reach the goal. On average it took 742 node samples each run, and it would have been longer had the planner been allowed to sample indefinitely until it found a path to the goal. Figure 10 is included to highlight how much more efficient the CL-RRT Glide algorithm is when expanding the search tree. Because of the cost heuristic, CL-RRT Glide does not expand infeasible nodes where the vehicle cannot reach the goal. This can be seen by the tree in (a) where algorithm only had to test a few nodes before it found a path to the goal. The unmodified CL-RRT algorithm wastes much more time trying to expand infeasible nodes, and its search tree in (b) is much larger before it eventually find a path to the goal. The planning time will likely grow with the size of the environment.



(a)



(b)

**Fig. 10** The figures show the difference between the expansion of the CL-RRT Glide (a) and unmodified CL-RRT (b) algorithms.

## V. Simulation

In order to ensure the feasibility of the samples to the control space from the CL-RRT algorithm, a realistic and computationally lightweight model of the vehicle dynamics is necessary. For the purposes of this research, the ZOHD Drift small fixed-wing UAV is used, shown in Fig. 11. The aircraft has a wingspan of 877 mm, a mass of 270 grams, and an airfoil similar to the AG24. A small brushless motor and lithium battery provide the thrust and an on board flight controller with a GPS provides communications, sensor feedback and control. The flight controller is compatible with the Arduplane open source flight control software, which allows new guidance controllers to be easily integrated into the Arduplane autonomy architecture.



**Fig. 11 ZOHD Drift Experimental Vehicle Platform**

**Table 3 ZOHD Drift Vehicle Attributes**

Attribute	Value
Wingspan	877 mm
Length	688 mm
Mass	270 grams
Airfoil	AG24

The core of the flight dynamics simulation is an accurate representation of the aerodynamic forces and moments acting on the aircraft. For the purposes of this work, a version of the panel method for approximating the aerodynamics of a fixed wing aircraft is used, similar to the approach used in [37]. The aircraft is assumed to be a rigid body that does not flex under the forces, which simplifies some of the force and moment equations. It is assumed that each lifting surface (left wing, right wing, horizontal and vertical stabilizers) of the aircraft is not affected by the presence of the other surfaces, and that only the lifting surfaces of the aircraft are capable of generating lift. The control surfaces are assumed to have local angle of attack and/or sideslip according to the angle of surface deflection. The wind angles ( $\alpha$  and  $\beta$ ) are averaged across each lifting surface, and the angular rates of the aircraft, as well as the overall airspeed, are considered when computing the relative velocities and wind angles of each lifting surface. The coefficients of lift, drag, and moment are interpolated with respect to the angle of attack or sideslip from XFOIL simulation data for the given airfoils of the lifting surfaces on the ZOHD Drift.

Equation 18 gives the formula for the dynamic pressure of the aircraft  $Q$ , which is proportional to the product of the air density,  $\rho$ , multiplied with the square of the airspeed,  $V_\infty$ .

$$Q = \frac{1}{2} \rho V_\infty^2 \quad (18)$$

The forces and moments acting on the wings are listed in Equations 19. The left and right lift coefficients  $C_{L_{wl}}$  and  $C_{L_{wr}}$ , drag coefficients  $C_{D_{wl}}$  and  $C_{D_{wr}}$ , and pitch moment coefficients  $C_{m_{wl}}$  and  $C_{m_{wr}}$  are interpolated from the Xfoil data for the AG24 at the respective angles of attack for each wing. The wing force and moment coefficients are also a function of the aileron deflection  $\delta_a$ , which is the output of the Roll PID controller in navigation controller from Fig. 2. The force coefficients are averaged and multiplied times  $Q$  and the wing area  $S_w$  to get the lift force  $L_w$  and drag force  $D_w$ . The roll moment  $\ell_w$  is computed from the difference in the lift forces between the left and right wings times a quarter of the wingspan  $b_w$  (the lift force is assumed to be acting halfway between the fuselage and the wingtip for each wing half). The pitch moment  $m_w$  is computed from the average pitch moment between the two wings times the mean aerodynamic chord  $c_{mac}$ ,  $Q$ , and  $S_w$  plus the lift and drag forces times their respective moment arms  $x_{AC_w}$  and  $z_{AC_w}$ .

$$\begin{aligned}
L_w &= -QS_w \frac{(C_{L_{wl}}(\alpha, \delta_a) + C_{L_{wr}}(\alpha, \delta_a))}{2} \\
D_w &= -QS_w \frac{(C_{D_{wl}}(\alpha, \delta_a) + C_{D_{wr}}(\alpha, \delta_a))}{2} \\
\ell_w &= QS_w \frac{b_w}{4} \frac{(C_{L_{wl}}(\alpha, \delta_a) - C_{L_{wr}}(\alpha, \delta_a))}{2} \\
m_w &= -QS_w c_{mac} \frac{(C_{m_{wl}}(\alpha, \delta_a) + C_{m_{wr}}(\alpha, \delta_a))}{2} + L_w x_{AC_w} \cos \alpha \\
&\quad + D_w (x_{AC_w} \sin \alpha + z_{AC_w} \cos \alpha)
\end{aligned} \tag{19}$$

The forces and moments acting on the horizontal stabilizer are listed in Equations 20. The lift coefficient  $C_{L_{hs}}$  and drag coefficient  $C_{D_{hs}}$  are interpolated from the Xfoil data for the symmetric airfoil NACA 0010 at the local angle of attack. The horizontal stabilizer force and moment coefficients are also a function of the elevator deflection  $\delta_e$ , which is the output of the Pitch PID controller in navigation controller from Fig. 2. The force coefficients are multiplied times  $Q$  and the horizontal stabilizer area  $S_{hs}$  to get the lift force  $L_{hs}$  and drag force  $D_{hs}$ . The pitch moment  $m_w$  acting on the aircraft is also computed from the forces times the respective moment arms  $x_{AC_{hs}}$  and  $z_{AC_{hs}}$  of the horizontal stabilizer aerodynamic center from the aircraft center of gravity.

$$\begin{aligned}
L_{hs} &= -QS_{hs} C_{L_{hs}}(\alpha, \delta_e) \\
D_{hs} &= -QS_{hs} C_{D_{hs}}(\alpha, \delta_e) \\
m_{hs} &= L_{hs} (x_{AC_{hs}} \cos \alpha + z_{AC_{hs}} \sin \alpha) \\
&\quad + D_{hs} (x_{AC_{hs}} \sin \alpha + z_{AC_{hs}} \cos \alpha)
\end{aligned} \tag{20}$$

The forces and moments acting on the vertical stabilizer are listed in Equations 21. The lift coefficient  $C_{Y_{vs}}$  and drag coefficient  $C_{D_{vs}}$  are interpolated from the Xfoil data for the symmetric airfoil NACA 0010 at the local angle of sideslip. The force coefficients are multiplied times  $Q$  and the vertical stabilizer area  $S_{vs}$  to get the side force  $Y_{vs}$  and drag force  $D_{vs}$ . The roll and yaw moments  $\ell_{vs}$  and  $n_{vs}$  acting on the aircraft are also computed from the forces times the respective

moment arms  $x_{AC_{vs}}$  and  $z_{AC_{vs}}$  of the vertical stabilizer aerodynamic center from the aircraft center of gravity.

$$\begin{aligned}
Y_{vs} &= -QS_{vs}C_{Y_{vs}}(\beta) \\
D_{vs} &= -QS_{vs}C_{D_{vs}}(\beta) \\
\ell_{vs} &= Y_{vs}z_{AC_{vs}}\cos\beta + D_{vs}z_{AC_{vs}}\sin\beta \\
n_{vs} &= -Y_{vs}x_{AC_{vs}}\cos\beta - D_{vs}x_{AC_{vs}}\sin\beta
\end{aligned} \tag{21}$$

While the fuselage is not a lifting surface and is not considered to have much effect on the aerodynamics of the aircraft, it can add a significant amount of drag. Because of that, a drag force  $D_f$  for the fuselage is computed from  $Q$  times the cross sectional area  $S_f$  and  $C_{D_f}$  to account for the inefficiencies caused by having a fuselage in Equation 22.

$$D_f = -QS_fC_{D_f} \tag{22}$$

Equations 23 sum and convert the wind axes forces components into body axes force components using the wind axes rotation matrix  $\mathbf{R}_{bW}$ .

$$\begin{aligned}
\mathbf{R}_{bW} &= \begin{bmatrix} \cos(\alpha)\cos(\beta) & -\cos(\alpha)\sin(\beta) & -\sin(\alpha) \\ \sin(\beta) & \cos(\beta) & 0 \\ \cos(\beta)\sin(\alpha) & -\sin(\alpha)\sin(\beta) & \cos(\alpha) \end{bmatrix} \\
\mathbf{F} = \mathbf{R}_{bW} \begin{bmatrix} D_w + D_{hs} + D_{vs} + D_f \\ Y_{vs} \\ L_w + L_{hs} \end{bmatrix}, \quad \mathbf{M} = \begin{bmatrix} \ell_w + \ell_{vs} \\ m_w + m_{hs} \\ n_{vs} \end{bmatrix}
\end{aligned} \tag{23}$$

The forces and moments are processed in a 6 DOF simulation of the aircraft dynamics using the Euler-Lagrange 2nd order method to predict the states of the next time step [38]. This method provides numerical stability and enables trajectory predictions with relatively large time steps. This is critical because the CL-RRT planning method relies on numerous forward simulations.

Strong agreement has been shown between the predicted closed loop flight paths of the simulated aircraft as compared to the real experimental ZOHD Drift. While the simplifying assumptions of the aerodynamic forces do reduce the accuracy, the reduced computational complexity enables rapid planning with small, low-power computers.

## VI. Experimental Setup

The glide planning algorithm was integrated into a small fixed wing autonomous aerial vehicle. This enables us to demonstrate the computational performance of the algorithm, the predictive performance of the simplified models, and the overall integration of the physical system. The glide path planning had to meet constraints of being fast enough to compute a path rapidly in real time after a loss of thrust in order to minimize altitude loss, and reliably choose a safe path for a successful landing.

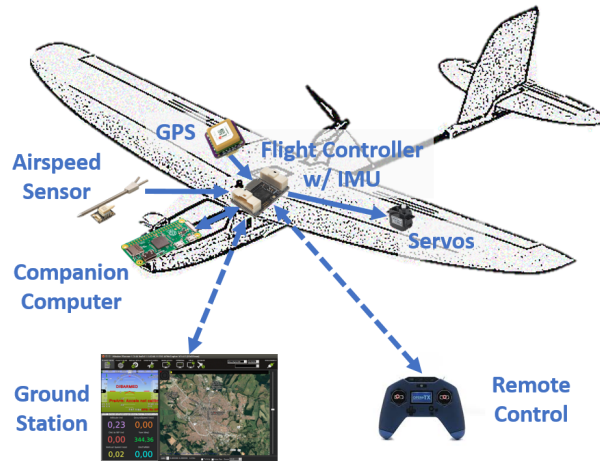
### A. Hardware Components

The hardware setup is assembled from the autonomous flight controller and sensor components available from mRobotics. The heart of the system is the Control Zero flight control board, which is a full feature flight controller arranged into a tiny package, perfect for fitting on board the ZOHD Drift. The flight controller has multiple IMU sensors with three axis accelerometers, gyros, magnetometers, and a barometer built in. A GPS receiver and airspeed sensor provides accurate three dimensional positioning, groundspeed, and windspeed velocity data. There are two radio connections to the flight controller, one for the remote control and one for the flight data and telemetry that is provided to the ground station. Additional computing capability is provided by a Raspberry Pi Zero, a miniature single board computer that fits inside the airplane. A diagram of the full hardware layout is shown in Fig. 12.

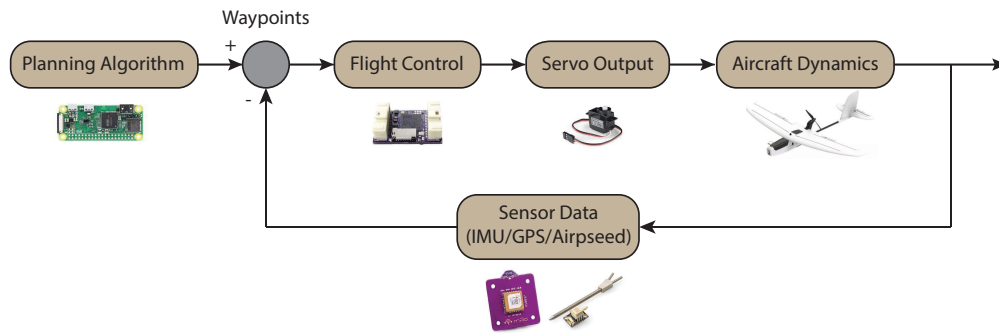
### B. Software Architecture

Our computing architecture consists of two computing systems. First, we have a low-level flight control system which performs feedback control of the aircraft based on information from the IMU and GPS system. We use a mRo Control Zero for the flight control hardware. The flight control software is based on the Arduplane open source flight autonomy project [35]. The Arduplane flight autonomy software already has extensive code to handle flight control, navigation, and communication tasks. This existing architecture was leveraged to create a custom flight control software library that uses our feedback control architecture. A new flight mode was created that replicates a loss of thrust emergency and switches the primary controller over to the glide controller.

In addition, we have a second, more powerful computer that focuses on path planning. This planning (guidance) computer communicates way-point reference input commands to the flight control computer. When the aircraft loses thrust, the planning computer is responsible for running the CL-RRT Glide algorithm to find a safe flight path to the approach waypoint and then uploading that flight path to the flight controller. In order to maximize the execution speed of the CL-RRT Glide algorithm, the code was written in Cython, a compiled version of Python that can be nearly as efficient as C++. The combination of simplified models, stable numerical solvers, and C++ implementation results in relatively rapid (4s) motion plan generation using a Raspberry Pi Zero single board computer (single core 1GHz processor).



(a)

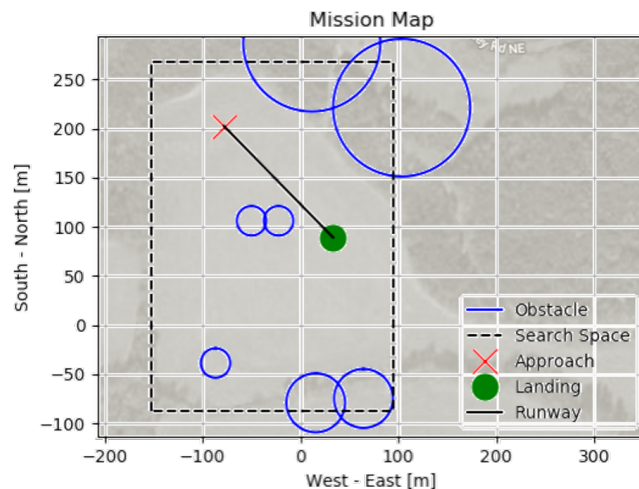


(b)

**Fig. 12** Diagram in (a) shows the layout of the flight autonomy hardware on board the ZOHD Drift experimental drone platform. The block diagram in (b) provides a visual representation of how the hardware components interact to control the aircraft.

### C. Experiment Plan

In order to test the CL-RRT Glide algorithm and full experimental system, a mission was designed for the flight field and virtual obstacle map shown in Fig. 13. The flight field chosen for the experiments has 75,000 sq meters of open space outside of Atlanta, GA. The virtual obstacles were added to the search space to create a more challenging environment for the path planner to navigate. A clear landing point was specified and an approach waypoint 150 meters downwind and 20 meters above ground level was set as the goal for the glide algorithm to reach for setting up a safe landing. This experiment design tests all aspects of autonomous loss-of thrust planning with the exception of four technical parts. The focus of the experiment is on the core technical contributions of this paper (motion planning and execution). First, we neglect failure detection because we assume a complete loss of thrust could be determined automatically by an autopilot. Second, we neglect perception of potential obstacles/no-go zones because it is outside the scope of this work. Obstacles or no-go zones are easily loaded in a pre-defined map and may only require accurate navigation (rather than perception with onboard sensors) to demonstrate the algorithm. Third, we neglect the process of identifying the best goal and assume it is already known. This may be a big assumption for small UAVs which would likely need to land in an unstructured environment due to lower altitudes and limited infrastructure. If multiple potential landing areas do exist, then they could be ranked using simple heuristics such as 2D Dubins distance. Alternatively, optimization methods could be used to first determine the best goal such as those described in [5]. Finally, we also are testing in favorable, low wind conditions because the current motion planning does not account for these random effects in the dynamics.



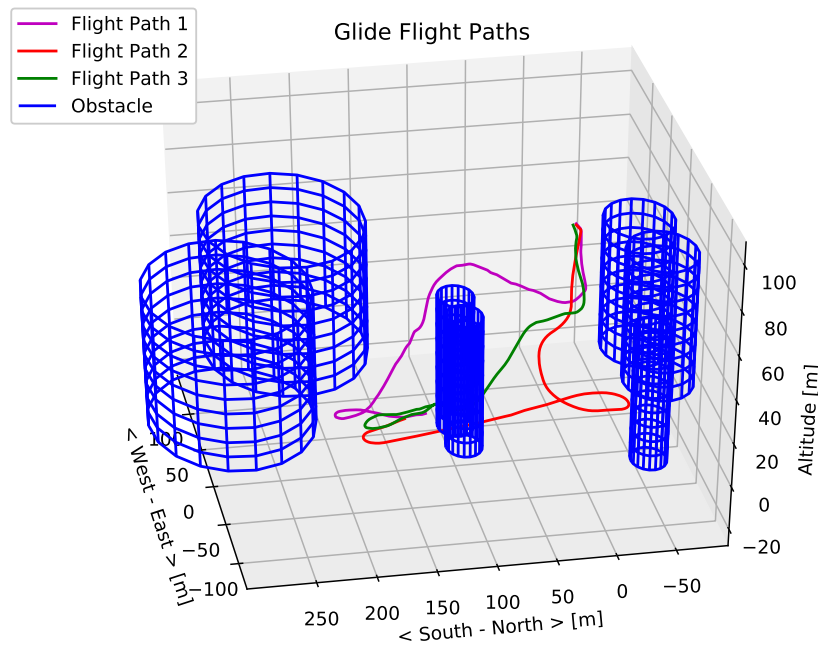
**Fig. 13** Mission map shows the same test environment overlaid onto a satellite image of the flying site.

The flight plan begins with a manual takeoff and then switches to an automatic circling while the airplane climbs to a mission altitude of 80 meters. Then the aircraft flies a way-point mission around the field, at which point it loses thrust at the far end of the field by switching to the glide flight mode. This triggers the CL-RRT Glide algorithm to begin

planning the flight path to the approach way-point. The algorithm begins from a contingency point representing where the aircraft will be after 5 seconds of flight in order to account for the time that the planner takes to find a path. The planner only returns a flight path that can feasibly reach the approach point within a certain threshold. The Dissipative TECS controls the aircraft at the demanded glide velocity for each intermediate waypoint, and for the final approach waypoint it dissipates the excess altitude. After reaching the approach waypoint, the aircraft glides along the runway until it lands.

## VII. Flight Data

The experimental flight system was successfully tested three times. Fig. 14 shows a three dimensional plot of the flight paths. For each flight, the path planner found a different route to the approach waypoint, reaching the desired position within 20m and the desired altitude within 5m each time. The paths also successfully avoided the virtual obstacles. Despite external disturbances from the wind, the closed loop controller was able to account for this unmodeled effect.



**Fig. 14** The plot shows the glide paths from each of the three experimental tests of the path planner. Each one was successful at finding a path that avoided all of the obstacles and reached the landing site.

Fig. 15 shows the flight path projected onto the horizontal plane along with the way-points from the CL-RRT glide path planner and the predicted glide path. The two dimensional view shows strong agreement between the predicted and actual flight paths, indicating that the simulated flight dynamics are a good predictor of the actual flight dynamics of the vehicle. The first way-point that the planner returned is the contingency point, or the point that the planner started from a few seconds ahead of where the aircraft was heading to account for the delay in planning. From there each way-point was

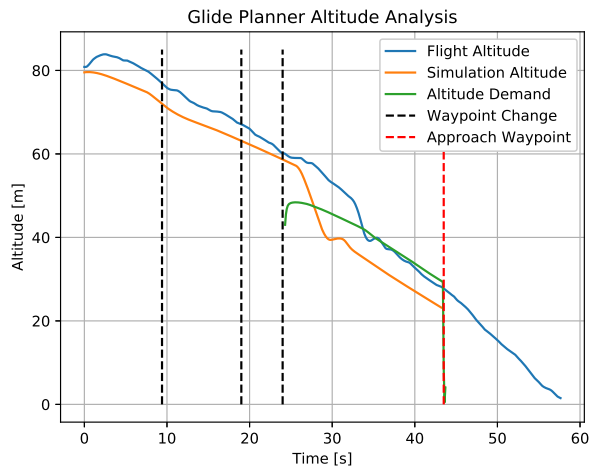
chosen by the planner to navigate around the obstacles and maintain enough altitude to reach the approach way-point.



**Fig. 15** The plot shows the full planning result from one of the path plans, along with the X-Y position of the flight path.

Fig. 16 shows the altitude control performance during the glide portion of the flight. The plot starts when the aircraft loses power and begins to enter a glide. The altitude initially increases because the aircraft is traveling slightly faster than the desired glide speed, so the aircraft gains some altitude as it approaches the demanded speed. From there, the glide descent follows a very similar trajectory to the predicted trajectory from the planner forward simulation. Once the aircraft starts heading towards the approach waypoint, the controller begins the process of correcting for the excess altitude. The controller accomplishes this by briefly increasing the aircraft velocity, thereby increasing the vehicle drag and the energy dissipation rate. As the error between the demanded and actual altitude decreases, the controller decreases the velocity and the aircraft resumes flying the desired glide path to reach the approach waypoint within 8 meters of the desired 20 meter altitude above ground level. The total horizontal path distance that the vehicle covered from the start to the goal approach point in this example is 345m. It is estimated from the glide slope that the aircraft could have flown a maximum horizontal distance of about 450m from the initial energy state of 60m altitude above, 215m away from the goal and a starting velocity of 12m/s. This is an ideal test that is simultaneously near the aircraft's glide range while still requiring the Dissipative TECS to account for the excess altitude before the aircraft reaches the goal.

These experimental tests demonstrate the performance and relevance of the planning algorithm for fixed wing aircraft. Multiple tests were performed and each time the planner was able to find a feasible path to the goal and execute an emergency glide landing. The time that the planning took was also manageable and did not prevent the aircraft from executing a safe path. The experimental results show that the paths returned by the planner, which were feasible for the dynamics model of the aircraft, could be successfully executed by the aircraft in real time. The controller was also able



**Fig. 16** The plot shows the altitude of the same gliding flight, showing both the actual aircraft altitude and the altitude predicted by the planner. The closed loop controller was able to correct the error.

to correct for the slight disturbances and errors during the flight to execute the path and avoid the obstacles in the search space.

## VIII. Conclusion

This work presented and experimentally demonstrated a new approach for loss-of-thrust aircraft motion planning. The proposed method builds on sampling-based planning with closed-loop prediction (CL-RRT), but adds new methods for evaluating nodes and executing spatial trajectories with finite energy. We showed that our methods provide improvements over a standard implementation of CL-RRT and provided experimental data on overall performance. The planner was shown to be successful operating in real time. Also, the glide controller was able to correct for excess altitude, and the control systems and models were able to handle real-world conditions. Several flights were performed with a loss of thrust event and a successfully planned and executed safe landing at a designated landing site. Wind is a particularly important factor that was not explored in this work, as well as identification of UAV landing sites in unstructured environments. While the experiments demonstrated good results, they were performed under reasonably benign conditions. We anticipate that small winds would be rejected by the onboard feedback control, but larger winds could substantially alter the vehicle trajectory. Feedback control could be complemented with feed-forward behaviors. For example, an onboard weather model that is used in the CL-RRT\* vehicle simulation predictions.

## IX. Acknowledgements

This work was supported by the Laboratory Directed Research and Development program at Sandia National Laboratories, a multimission laboratory managed and operated by the National Technology and Engineering Solutions of

Sandia LLC, a wholly owned subsidiary of Honeywell International Inc. for the U.S. Department of Energy’s National Nuclear Security Administration under contract DE-NA0003525. This paper describes objective technical results and analysis. Any subjective views or opinions that might be expressed in the paper do not necessarily represent the views of the U.S. Department of Energy or the United States Government.

## References

- [1] “Aircraft Accident Report: Loss of Thrust in Both Engines After Encountering a Flock of Birds and Subsequent Ditching on the Hudson River, US Airways Flight 1549, Airbus A320-214, N106US, Weehawken, New Jersey, January 15, 2009,” 2010. URL <https://trid.trb.org/view/917904>, number: NTSB/AAR-10/03.
- [2] Atkins, E., “Emergency Landing Automation Aids: An Evaluation Inspired by US Airways Flight 1549,” *AIAA Infotech@Aerospace 2010*, Infotech@Aerospace Conferences, American Institute of Aeronautics and Astronautics, 2010. <https://doi.org/10.2514/6.2010-3381>, URL <https://arc.aiaa.org/doi/10.2514/6.2010-3381>.
- [3] Grauer, J. A., “Aircraft Fault Detection Using Real-Time Frequency Response Estimation,” *AIAA Guidance, Navigation, and Control Conference*, American Institute of Aeronautics and Astronautics, 2016. <https://doi.org/10.2514/6.2016-0372>, URL <https://arc.aiaa.org/doi/abs/10.2514/6.2016-0372>, \_eprint: <https://arc.aiaa.org/doi/pdf/10.2514/6.2016-0372>.
- [4] Gopisetty, S., and Stengel, R., “Detecting and identifying multiple failures in a flight control system,” *Guidance, Navigation, and Control Conference and Exhibit*, American Institute of Aeronautics and Astronautics, 2012. <https://doi.org/10.2514/6.1998-4488>, URL <https://arc.aiaa.org/doi/abs/10.2514/6.1998-4488>, \_eprint: <https://arc.aiaa.org/doi/pdf/10.2514/6.1998-4488>.
- [5] Atkins, E. M., Portillo, I. A., and Strube, M. J., “Emergency Flight Planning Applied to Total Loss of Thrust,” *Journal of Aircraft*, Vol. 43, No. 4, 2006, pp. 1205–1216. <https://doi.org/10.2514/1.18816>, URL <https://doi.org/10.2514/1.18816>, publisher: American Institute of Aeronautics and Astronautics \_eprint: <https://doi.org/10.2514/1.18816>.
- [6] Wu, H., and Mora-Camino, F., “Glide Control for Engine-Out Aircraft,” *AIAA Guidance, Navigation, and Control Conference*, American Institute of Aeronautics and Astronautics, Minneapolis, Minnesota, 2012. <https://doi.org/10.2514/6.2012-4442>, URL <https://arc.aiaa.org/doi/10.2514/6.2012-4442>.
- [7] Tabor, S., Guilliard, I., and Kolobov, A., “ArduSoar: an Open-Source Thermalling Controller for Resource-Constrained Autopilots,” *arXiv:1802.08215 [cs]*, 2018. URL <http://arxiv.org/abs/1802.08215>, arXiv: 1802.08215.
- [8] Chakrabarty, A., and Langelaan, J., “Energy Maps for Long-Range Path Planning for Small- and Micro- UAVs,” *AIAA Guidance, Navigation, and Control Conference*, American Institute of Aeronautics and Astronautics, Chicago, Illinois, 2009. <https://doi.org/10.2514/6.2009-6113>, URL <https://arc.aiaa.org/doi/10.2514/6.2009-6113>.
- [9] Paul, S., Hole, F., Zyteck, A., and Varela, C. A., “Flight Trajectory Planning for Fixed-Wing Aircraft in Loss of Thrust Emergencies,” *arXiv:1711.00716 [cs]*, 2017. URL <http://arxiv.org/abs/1711.00716>, arXiv: 1711.00716.

- [10] Adler, A., Bar-Gill, A., and Shimkin, N., “Optimal flight paths for engine-out emergency landing,” *2012 24th Chinese Control and Decision Conference (CCDC)*, 2012, pp. 2908–2915. <https://doi.org/10.1109/CCDC.2012.6244461>, iISSN: 1948-9447.
- [11] Lin, C. E., and Shao, P.-C., “Failure analysis for an unmanned aerial vehicle using safe path planning,” *Journal of Aerospace Information Systems*, Vol. 17, No. 7, 2020, pp. 358–369.
- [12] Kuwata, Y., Teo, J., Karaman, S., Fiore, G., Frazzoli, E., and How, J., “Motion Planning in Complex Environments Using Closed-loop Prediction,” *AIAA Guidance, Navigation and Control Conference and Exhibit*, American Institute of Aeronautics and Astronautics, Honolulu, Hawaii, 2008. <https://doi.org/10.2514/6.2008-7166>, URL <https://arc.aiaa.org/doi/10.2514/6.2008-7166>.
- [13] Hart, P. E., Nilsson, N. J., and Raphael, B., “A Formal Basis for the Heuristic Determination of Minimum Cost Paths,” *IEEE Transactions on Systems Science and Cybernetics*, Vol. 4, No. 2, 1968, pp. 100–107. <https://doi.org/10.1109/TSSC.1968.300136>, conference Name: IEEE Transactions on Systems Science and Cybernetics.
- [14] Daniel, K., Nash, A., Koenig, S., and Felner, A., “Theta\*: Any-Angle Path Planning on Grids,” *Journal of Artificial Intelligence Research*, Vol. 39, 2010, pp. 533–579. <https://doi.org/10.1613/jair.2994>, URL <https://jair.org/index.php/jair/article/view/10676>.
- [15] Karaman, S., and Frazzoli, E., “Sampling-based Algorithms for Optimal Motion Planning,” *arXiv:1105.1186 [cs]*, 2011. URL <http://arxiv.org/abs/1105.1186>, arXiv: 1105.1186.
- [16] LaValle, S., “Rapidly-exploring random trees : a new tool for path planning,” *undefined*, 1998. URL <http://msl.cs.illinois.edu/~laval/papers/Lav98c.pdf>.
- [17] Arslan, O., and Tsiotras, P., “Use of relaxation methods in sampling-based algorithms for optimal motion planning,” *2013 IEEE International Conference on Robotics and Automation*, IEEE, Karlsruhe, Germany, 2013, pp. 2421–2428. <https://doi.org/10.1109/ICRA.2013.6630906>, URL <http://ieeexplore.ieee.org/document/6630906/>.
- [18] Kuwata, Y., Teo, J., Fiore, G., Karaman, S., Frazzoli, E., and How, J. P., “Real-time motion planning with applications to autonomous urban driving,” *IEEE Transactions on control systems technology*, Vol. 17, No. 5, 2009, pp. 1105–1118.
- [19] Frazzoli, E., Dahleh, M. A., and Feron, E., “Real-Time Motion Planning for Agile Autonomous Vehicles,” *Journal of Guidance, Control, and Dynamics*, Vol. 25, No. 1, 2002, pp. 116–129. <https://doi.org/10.2514/2.4856>, URL <https://doi.org/10.2514/2.4856>, publisher: American Institute of Aeronautics and Astronautics \_eprint: <https://doi.org/10.2514/2.4856>.
- [20] Levin, J. M., Nahon, M., and Paranjape, A. A., “Real-time motion planning with a fixed-wing UAV using an agile maneuver space,” *Autonomous Robots*, Vol. 43, No. 8, 2019, pp. 2111–2130. <https://doi.org/10.1007/s10514-019-09863-2>, URL <https://doi.org/10.1007/s10514-019-09863-2>.
- [21] Ramana, M. V., Varma, S. A., and Kothari, M., “Motion Planning for a Fixed-Wing UAV in Urban Environments,” *IFAC-PapersOnLine*, Vol. 49, No. 1, 2016, pp. 419–424. <https://doi.org/10.1016/j.ifacol.2016.03.090>, URL <https://www.sciencedirect.com/science/article/pii/S2405896316300908>.

- [22] Asadi, D., Sabzehparvar, M., Atkins, E. M., and Talebi, H. A., “Damaged Airplane Trajectory Planning Based on Flight Envelope and Motion Primitives,” *Journal of Aircraft*, Vol. 51, No. 6, 2014, pp. 1740–1757. <https://doi.org/10.2514/1.C032422>, URL <https://doi.org/10.2514/1.C032422>, publisher: American Institute of Aeronautics and Astronautics \_eprint: <https://doi.org/10.2514/1.C032422>.
- [23] Arslan, O., Berntorp, K., and Tsiotras, P., “Sampling-based Algorithms for Optimal Motion Planning Using Closed-loop Prediction,” *arXiv:1601.06326 [cs]*, 2016. URL <http://arxiv.org/abs/1601.06326>, arXiv: 1601.06326.
- [24] Lawrance, N., and Sukkarieh, S., “Wind Energy Based Path Planning for a Small Gliding Unmanned Aerial Vehicle,” *AIAA Guidance, Navigation, and Control Conference*, Guidance, Navigation, and Control and Co-located Conferences, American Institute of Aeronautics and Astronautics, 2009. <https://doi.org/10.2514/6.2009-6112>, URL <https://arc.aiaa.org/doi/10.2514/6.2009-6112>.
- [25] Kothari, M., and Postlethwaite, I., “A Probabilistically Robust Path Planning Algorithm for UAVs Using Rapidly-Exploring Random Trees,” *J. Intell. Robotic Syst.*, 2013. <https://doi.org/10.1007/s10846-012-9776-4>.
- [26] Luders, B. D., Sugel, I., and How, J. P., “Robust Trajectory Planning for Autonomous Parafoils under Wind Uncertainty,” *AIAA Infotech@Aerospace (I@A) Conference*, American Institute of Aeronautics and Astronautics, Boston, MA, 2013. <https://doi.org/10.2514/6.2013-4584>, URL <https://arc.aiaa.org/doi/10.2514/6.2013-4584>.
- [27] Lin, Y., and Saripalli, S., “Sense and avoid for unmanned aerial vehicles using ADS-B,” *2015 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2015, pp. 6402–6407.
- [28] Lin, Y., and Saripalli, S., “Sampling-based path planning for UAV collision avoidance,” *IEEE Transactions on Intelligent Transportation Systems*, Vol. 18, No. 11, 2017, pp. 3179–3192.
- [29] Coombes, M., Chen, W.-H., and Render, P., “Landing Site Reachability in a Forced Landing of Unmanned Aircraft in Wind,” *Journal of Aircraft*, Vol. 54, No. 4, 2017, pp. 1415–1427. <https://doi.org/10.2514/1.C033856>, URL <https://doi.org/10.2514/1.C033856>, publisher: American Institute of Aeronautics and Astronautics \_eprint: <https://doi.org/10.2514/1.C033856>.
- [30] Di Donato, P. F. A., and Atkins, E. M., “Optimizing Steady Turns for Gliding Trajectories,” *Journal of Guidance, Control, and Dynamics*, Vol. 39, No. 12, 2016, pp. 2627–2637. <https://doi.org/10.2514/1.G000319>, URL <https://arc.aiaa.org/doi/10.2514/1.G000319>, publisher: American Institute of Aeronautics and Astronautics.
- [31] Almozel, A., Feron, E. M., Saber, S. I., Cloiseau, C., and Vanderverter, K., “Safe Trajectory Planning for Safety Critical Drone Delivery,” *AIAA AVIATION 2023 Forum*, American Institute of Aeronautics and Astronautics, 2023. <https://doi.org/10.2514/6.2023-3547>, URL <https://arc.aiaa.org/doi/abs/10.2514/6.2023-3547>, \_eprint: <https://arc.aiaa.org/doi/pdf/10.2514/6.2023-3547>.
- [32] Fang, X., Wan, N., Jafarnejadsani, H., Sun, D., Holzapfel, F., and Hovakimyan, N., “Emergency Landing Trajectory Optimization for Fixed-Wing UAV under Engine Failure,” *AIAA Scitech 2019 Forum*, American Institute of Aeronautics and Astronautics, San Diego, California, 2019. <https://doi.org/10.2514/6.2019-0959>, URL <https://arc.aiaa.org/doi/10.2514/6.2019-0959>.

- [33] Li, P., Chen, X., and Li, C., “Emergency landing control technology for UAV,” *Proceedings of 2014 IEEE Chinese Guidance, Navigation and Control Conference*, 2014, pp. 2359–2362. <https://doi.org/10.1109/CGNCC.2014.7007537>.
- [34] Park, S., Deyst, J., and How, J., “A New Nonlinear Guidance Logic for Trajectory Tracking,” *AIAA Guidance, Navigation, and Control Conference and Exhibit*, American Institute of Aeronautics and Astronautics, Providence, Rhode Island, 2004. <https://doi.org/10.2514/6.2004-4900>, URL <http://arc.aiaa.org/doi/10.2514/6.2004-4900>.
- [35] “Arduplane project,” <https://ardupilot.org/plane/index.html>, 2009. Accessed: 2022-02-15.
- [36] Argyle, M. E., and Beard, R. W., “Nonlinear Total Energy Control for the Longitudinal dynamics of an aircraft,” *2016 American Control Conference (ACC)*, IEEE, Boston, MA, USA, 2016, pp. 6741–6746. <https://doi.org/10.1109/ACC.2016.7526733>, URL <http://ieeexplore.ieee.org/document/7526733/>.
- [37] Khan, W., and Nahon, M., “Real-time modeling of agile fixed-wing UAV aerodynamics,” *2015 International Conference on Unmanned Aircraft Systems (ICUAS)*, 2015, pp. 1188–1195. <https://doi.org/10.1109/ICUAS.2015.7152411>.
- [38] Lee, T., “Computational Geometric Mechanics and Control of Rigid Bodies.” Thesis, 2008. URL <http://deepblue.lib.umich.edu/handle/2027.42/60804>, accepted: 2008-08-25T20:55:39Z.