ENERGY-AWARE PATH PLANNING FOR FIXED-WING SEAPLANE UAVS

A Thesis
Submitted to the Faculty
in partial fulfillment of the requirements for the
degree of

Master of Science

in

Computer Science

by Ben Wolsieffer

Guarini School of Graduate and Advanced Studies
Dartmouth College
Hanover, New Hampshire

August, 2023

Examining Committee:

(chair) Alberto Quattrini Li

Adithya Pediredla

Laura Ray

F. Jon Kull, Ph.D.
Dean of the Guarini School of Graduate and Advanced Studies

Abstract

Fixed-wing unmanned aerial vehicles (UAVs) are commonly used for remote sensing applications over water bodies, such as monitoring water quality or tracking harmful algal blooms. However, there are some types of measurements that are difficult to accurately obtain from the air. In existing work, water samples have been collected in situ either by hand, with an unmanned surface vehicle (USV), or with a vertical takeoff and landing (VTOL) UAV such as a multirotor. We propose a path planner, landing control algorithm, and energy estimator that will allow a low-cost and energy efficient fixed-wing UAV to carry out a combined remote sensing and direct water sampling mission without requiring sophisticated sensors and using limited onboard computation. Finally, we demonstrate a fully autonomous mission on a modified off-the-shelf RC aircraft. The aircraft flies a survey pattern, lands at a series of sampling points and then returns to the starting location while respecting the available energy budget. In our experiments, we completed multiple sampling missions in the real world with no aborted landings or crashes and an overall energy estimation error of approximately 5%.

Preface

Thank you to my advisor, Alberto Quattrini Li, for your help and advice throughout the project, and particularly for your assistance with running and documenting the experiments. Thank you as well to the rest of my committee, Adithya Pediredla and Laura Ray, for your valuable feedback that helped me improve my thesis.

Thank you to my parents for your support and invaluable help editing this thesis.

I greatly appreciate the financial assistance I received to attend Dartmouth, including my graduate tuition scholarship and the Krehbiel scholarship that helped support my undergraduate education.

Finally, thank you to the Dartmouth College computer science department and the Thayer School of Engineering for the opportunities, knowledge and overall great experience they have provided throughout my time as an undergraduate and graduate student.

Contents

Chapter 1

Introduction

In this thesis, we aim to develop and test an energy-aware path planner to allow a fixed-wing unmanned aerial vehicle (UAV) to conduct a fully autonomous water sampling mission. UAVs are commonly used for remote sensing. Remote sensing cameras carried by UAVs can quickly provide coverage of large areas. UAVs provide both higher spatial and temporal resolution than satellites, while also having far lower costs and increased flexibility compared to manned aircraft. In particular, UAV based remote sensing has been applied to monitor harmful algal blooms (HABs) and track water quality [1], [2]. However, some information can only be obtained through direct sampling of the water. Traditionally, water samples have been made by hand, and more recently unmanned surface vehicles have been applied to the task, such as in [3]. Others have started to use UAVs for water sampling as well, but most existing work uses vertical-takeoff and landing (VTOL) UAVs such as multirotors [4], [5]. VTOL aircraft are easy to deploy and control because they do not require large areas to takeoff and land and can follow arbitrary three-dimensional paths. This flexibility comes at the expense of efficiency. A VTOL aircraft requires more power than a fixed-wing aircraft to carry the same payload, resulting in a shorter range. For example, the DJI Mavic 3 quadcopter has a maximum flight time of 46 minutes with the built-in camera payload, while the AgEagle (senseFly) eBee X fixed-wing UAV has a maximum flight time of 90 minutes with a mapping camera payload. In addition, fixed-wing aircraft are mechanically simpler and less expensive. This lower efficiency can be partially mitigated by using a helicopter or a hybrid fixed-wing/multirotor design, commonly known as a quadplane, but both types of aircraft are complex and expensive.

Pure fixed-wing aircraft are mechanically simple, fast, efficient and low-cost. A

Figure 1.1: Image of our experimental aircraft in flight during a snowstorm.

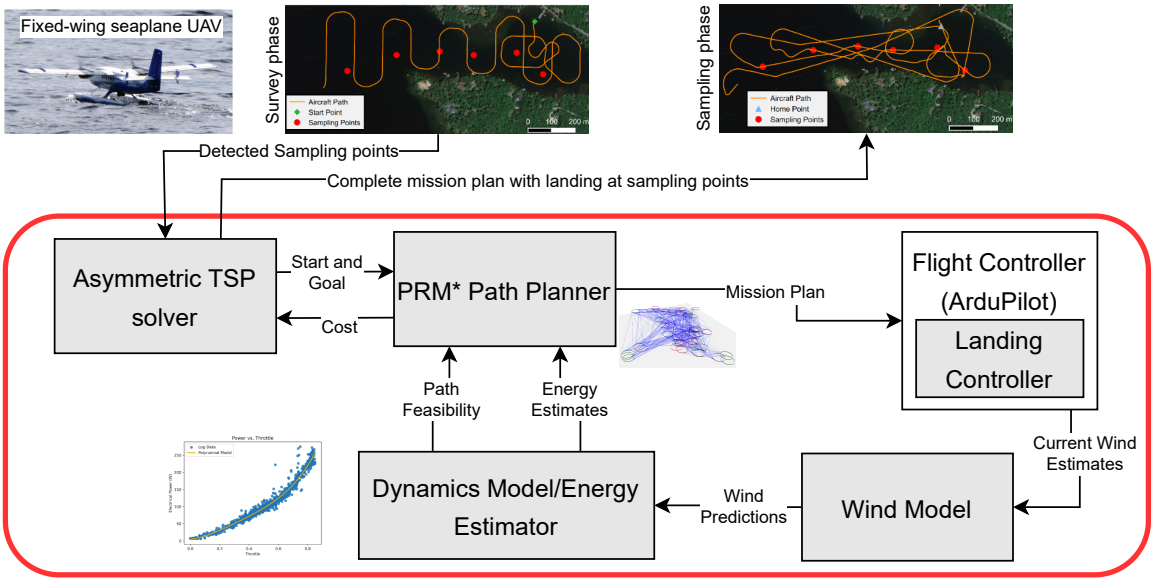


Figure 1.2: Proposed pipeline for finding energy-efficient plans with landings and takeoffs for a fixed-wing UAV.

wide variety of designs are produced commercially as remote control (RC) aircraft, and these can be adapted into UAVs with low-cost hardware and open source software. On the other hand, the flight dynamics of fixed-wing UAVs place constraints on the paths they can follow and increase the pilot skill required to operate them. Our approach solves these problems by providing a framework to enable fully autonomous remote sensing and direct sampling missions with a single vehicle. We propose an asymptotically optimal PRM* based path planner that generates energy efficient paths while respecting fixed-wing dynamics constraints in the presence of static obstacles. Additionally, we develop a reliable landing control technique that does not require an airspeed sensor or altitude rangefinder. We converted an off-the-shelf RC floatplane to a water landing UAV (see Figure 1.1), successfully demonstrated online planning, and executed a series of fully autonomous takeoffs, flights and landings on a lake to simulate a water sampling mission. Our work lays the foundation for more capable and lower cost environmental monitoring tools. A high level schematic of our work is shown in Figure 1.2.

Chapter 2

Related Work

Other researchers have applied various types of UAVs for both remote sensing and direct sampling of water bodies. A traditional approach for remote sensing is to use a fixed-wing aircraft equipped with a camera as is done in [6]. [1] and [2] review seven applications of fixed-wing aircraft specifically to remote sensing of harmful algal blooms. Most of these use flying wing style airframes equipped with RGB and near-infared cameras. Two use more exotic sensors: a hyperspectral camera and a spectroradiometer. VTOL aircraft have also been used for remote sensing. A quadcopter equipped with a multispectral camera is described in [7], while [1] and [2] additionally review several papers that rely on different types of multirotors and helicopters for remote sampling. [8] applies a less common type of aircraft, a tethered balloon. [2] discusses the tradeoffs between different kinds of aircraft for remote sensing, noting that fixed-wing aircraft have longer flight times than multirotors or helicopters but are more difficult to operate because they normally need large open areas adjacent to the water body for takeoff and landing. Our work aims to overcome the limitations of fixed-wing aircraft through increased autonomy.

For water sampling tasks, existing work primarily uses VTOL capable rotorcraft. A common approach is to dangle a water collection device from the bottom of the aircraft and collect data while hovering. Both [4] and [5] use a hexacopter in this manner, while [9] uses a helicopter. As in our work, one alternative is to land on the water while collecting samples [10]. Rather than a water collection device, [11] uses a dangling sensor array deployed from a hexacopter while hovering to measure water temperatures at different depths.

Outside of work designed specifically for HAB monitoring, a wide range of designs for amphibious UAVs have been proposed. Many models of flying-boat or floatplane

RC aircraft are commercially available and can be adapted as UAVs. A few aircraft, such as the AeroMapper Talon (used in [12]) and SELab Hamadori [13], are designed as amphibious UAVs; however, the AeroMapper Talon can land on water but cannot take off again. Our work targets these traditional style aircraft; however, others have proposed novel aircraft designs for aquatic environments. [14], [15] and [16] propose designs for fixed-wing aircraft that can operate underwater as well as in the air. These vehicles land by splashing down into the water and accelerate underwater while taking off. This requires a high thrust to weight ratio and a robust structure, which limits payload capacity but reduces the need for precise control. [17] develops a submersible quadcopter that uses its flight motors for propulsion and control underwater in addition to a variable buoyancy system. Some add flotation to traditional VTOL aircraft designs to allow water landings, as was done in [10]. Another such design is a hexacopter with a foam frame/propeller guard to allow it to float [18]. This aircraft is partially submerged while landed, requiring full waterproofing of its electronics. [19] uses a unique design of a tilt-rotor fixed-wing/multirotor hybrid that includes flotation in the tilting motor frame to achieve flotation. [20] and [21] describe designs for a multirotor aircraft attached to a hovercraft to allow landing and maneuvering on the water surface.

Some researchers have developed floatplane or flying boat style fixed-wing UAVs. [22] describes the "Flying Fish," a custom built solar powered floatplane designed to drift on the ocean and make short flights to reposition itself. This aircraft was designed to withstand ocean waves, but this increased its weight and limited its flight time. The authors discuss the numerous issues they faced with fouling of airspeed sensors. They added pitot heaters and a redundant pitot tube, while also developing an algorithm to reject measurements from fouled or faulty sensors [23]. The authors also implement a landing control technique that relies on an ultrasonic rangefinder to trigger the flare at the correct altitude above the water surface. [24] describes a twin-hull flying boat, with the added unique capability of folding the wings to act as sails. This allows efficient propulsion on the water surface. Autonomous takeoff and landing are implemented using the standard functions provided by the PX4 flight controller firmware. [25] describes another twin-hull flying boat, with a simple landing controller that maintains a constant airspeed until touchdown by controlling pitch. Lastly, [26] describes a conceptual design for a solar powered flying boat, which is designed to fly during the day and drift on the water surface at night. Most of these projects rely on an airspeed sensors, and in particular the issues faced by the "Flying

Fish" project motivated us to avoid this requirement in our work.

A wide variety of planning algorithms have been developed for fixed-wing UAVs and other types of aircraft. [27] describes a A* based path planner that uses an energy metric based on a simplified aircraft physics model, neglecting wind and energy consumed during turns. Additionally, this planner does not incorporate takeoffs or landings. Another energy aware planner was developed for the "Flying Fish" aircraft [28]. This planner supports multiple goals and generates Dubins paths between each pair of goals. The planner solves a traveling salesman problem (TSP) to choose an ordering of goals. The planner considers finite height cylindrical obstacles. If the Dubins path between a pair of goals collides with an obstacle, this path is not included in the TSP graph. Solving the TSP is complicated by the presence of negative costs due to energy recovery with solar panels. The energy model is based on a second-order polynomial mapping from throttle to power. Average throttle values measured during different flight segments are used to estimate the average power during each segment, and the segment durations were estimated by calculating the groundspeed in the presence of wind. This energy model is similar to ours but does not incorporate a dynamics model or support spatially varying wind fields. [29] proposes a multi-goal planner to solve an orienteering problem with a fixed-wing aircraft. The planner attempts to maximize the reward obtained from visiting goals while subject to a limited travel budget. A Dubins path is generated between each goal. The Dubins path may also incorporate a climb or descent, subject to climb angle limits. If these limits are exceeded, spiral turns are added to allow time for the aircraft to climb or descend. [30] develops an RRT based planner for an autonomous helicopter. Paths generated by the planner are then smoothed with a cubic Bezier spline.

Other researchers have developed energy models for UAVs, independent of a path planner. [31] describes an energy model for a quadcopter, derived from energy measurements under a variety of controlled conditions. These measurements were used to fit polynomial models that describe energy usage under various flight states, such as takeoff, flying horizontally, flying vertically or hovering. [32] proposes an energy model for a fixed-wing aircraft based on a Fourier series regression. Such a regression is periodic, so this model is only applicable to missions with repetitive segments, or phases with fixed lengths such as takeoff or landing. The model cannot be used to predict the energy of a new mission under arbitrary wind conditions, as required in our application. Lastly, [33] develops an instantaneous power model derived from the physical dynamics of a fixed-wing aircraft. Predicting the power with this model

requires full knowledge of the current aircraft state as well as detailed characterization of the aircraft's aerodynamic parameters. This characterization requires a wind tunnel, while our model can be tuned using flight data only.

Chapter 3

Overview

We consider a fixed-wing UAV that can takeoff, land and fly in a 3D environment, while constrained by aircraft dynamics, wind conditions and an energy budget. The environment may contain known static obstacles, which could be extracted from satellite imagery. Our proposed system executes a mission split into two phases: (1) the survey phase, during which the sampling points are identified, and (2) the sampling phase, which starts when the predicted energy required to visit the sampling points equals the remaining energy in the battery, minus a safety margin. This work focuses on the sampling phase. We assume that a survey coverage pattern is given and there exists a way to detect sampling points during the survey phase. Given the sampling points, the current wind conditions and a map of obstacles as inputs, our algorithm generates a path to land at each of the sampling points while minimizing the total energy consumed. This approach serves to demonstrate how each of our proposed techniques can work together, but the individual components could serve as the building blocks of other types of missions for specific applications.

We performed experiments to validate each component both in simulation and using a small, fixed-wing UAV. The UAV uses an embedded Linux computer to run the planner and mission controller, while a flight controller running the open source ArduPilot firmware [34] performs real-time control, including executing the landing algorithm.

The following chapters describe in greater detail the path planner, landing control algorithm, energy estimator and demonstration mission controller. For ease of reference, Table 3.1 includes symbols that are used throughout the thesis.

Table 3.1: Nomenclature.

$a_{takeoff}$	=	average acceleration during the beginning of takeoff
E	=	energy
h_{in}	=	unfiltered target altitude
h_0	=	first altitude filter output
h_{out}	=	final filtered target altitude
h_{flare}	=	landing flare altitude
$h_{takeoff}$	=	takeoff target altitude
k_{roll}	=	roll to throttle gain
P	=	power
P_{flare}	=	average power during landing flare
$P_{takeoff}$	=	average power during takeoff
r	=	turn radius
t	=	time
T	=	throttle
T_{cruise}	=	throttle to maintain cruise airspeed in level flight
T_{max}	=	maximum throttle
T_{min}	=	minimum throttle
v_a	=	airspeed
$v_{a,app}$	=	airspeed during landing approach
$v_{a,cruise}$	=	cruise airspeed in level flight
$v_{a,max}$	=	airspeed at maximum descent rate
$v_{a,takeoff}$	=	airspeed during takeoff
v_c	=	climb rate
$v_{c,max}$	=	maximum demanded climb rate
$v_{c,min}$	=	minimum demanded climb rate (negative)
$v_{c,flare}$	=	climb rate during landing flare (negative)
$v_{c,takeoff}$	=	average climb rate during takeoff
v_g	=	groundspeed
$v_{g,3d}$	=	groundspeed including vertical component
w_x, w_y	=	wind vector
x, y	=	position in local coordinates
γ	=	pitch angle
γ_0	=	pitch angle to maintain level flight at cruise

γ_{max}	=	maximum demanded pitch angle
γ_{min}	=	minimum demanded pitch angle (negative)
ϕ	=	roll angle
θ	=	heading angle

Chapter 4

Path Planning

We develop an asymptotically optimal probabilistic roadmap (PRM*) [35] based path planner to allow the aircraft to navigate in the presence of static obstacles. The planner generates energy optimal paths using the energy estimator as a cost function. The planner roadmap is constructed at the beginning of the flight (and could be generated offline if necessary) and is used to answer many planning queries over the course of the survey and landings. Obstacles are represented as 2D polygons with an optional associated minimum altitude constraint. The aircraft altitude must be above the specified minimum altitude while inside the obstacle boundary polygon. Obstacles may also be inverted, which means the minimum altitude constraint must be satisfied while the aircraft is outside the boundary polygon. If no minimum altitude is specified, the obstacle is assumed to be infinitely tall. There must exist at least one inverted obstacle with no minimum altitude, which defines the outer boundary of the navigation area. If there are multiple such obstacles, the intersection of their boundary polygons defines the navigation area. The rectangular bounding box of the navigation area defines the horizontal position sampling domain for the path planner. Where there are no defined obstacles, the ground is assumed to be flat and located at zero altitude relative to the aircraft's starting position. This assumption normally holds while operating over a water body. In our implementation, obstacles are represented using the GeoJSON format [36]. Each obstacle is represented by a polygon feature contained in a top-level feature collection. Each obstacle feature has properties to specify the minimum altitude and whether it is inverted. In GeoJSON format, obstacle polygon vertices are specified in WGS84 latitude/longitude coordinates, but planning is done in a local tangent plane coordinate frame using a spherical Earth approximation. Given a WGS84 coordinate (λ, ϕ) and origin coordi-

nate (λ_0, ϕ_0), the local easting coordinate x and northing coordinate y are given by Equations 4.1 and 4.2, respectively. A spherical approximation is used rather than the more accurate ellipsoidal approximation in order to match the flight controller's internal local coordinate calculations.

$$x = k_\lambda \cos\lambda(\phi - \phi_0) \tag{4.1}$$

$$y = k_\lambda(\lambda - \lambda_0) \tag{4.2}$$

where

$$k_\lambda = \frac{6378100\,\mathrm{m} \cdot \pi}{180\,\mathrm{deg}} \tag{4.3}$$

Plans are constructed from a set of motion primitives, each of which corresponds to a type of waypoint supported by the flight controller. The motion primitives are takeoff, landing, transit, and turn. Transits are straight line segments corresponding to simple waypoints placed at the end of the transit leg. The start and end of a transit may be at different altitudes, subject to limits on the maximum climb or sink rate of the aircraft. Turn primitives are modeled as constant altitude, constant radius turns and are implemented as circular loiters, where the planner guarantees that the end of the previous transit and the start of the next transit are both tangent to the loiter circle. The flight controller is configured to exit the loiter along the tangent when the aircraft is pointing toward the next waypoint.

In most circumstances, all planned paths start from either a midair loiter or a takeoff and consist of an alternating sequence of transits and turns, followed by a landing. By using loiter turns to transition between each segment of the mission, we can precisely predict the path of the aircraft and be confident that it will not collide with obstacles. The aircraft may deviate slightly from the predicted path due to external disturbances and the transient behavior of the aircraft as it transitions from straight to turning flight and vice versa, but these effects tend to be small in practice. If the mission were to consist only of straight transit segments with no loiter turns in between, the behavior of the aircraft while turning would be highly dependent on the tuning of the flight controller control loops and would be difficult to model. On the other hand, when using loiters as constant radius turns, we cannot guarantee that the flight controller will trigger the aircraft to immediately exit the loiter upon reaching

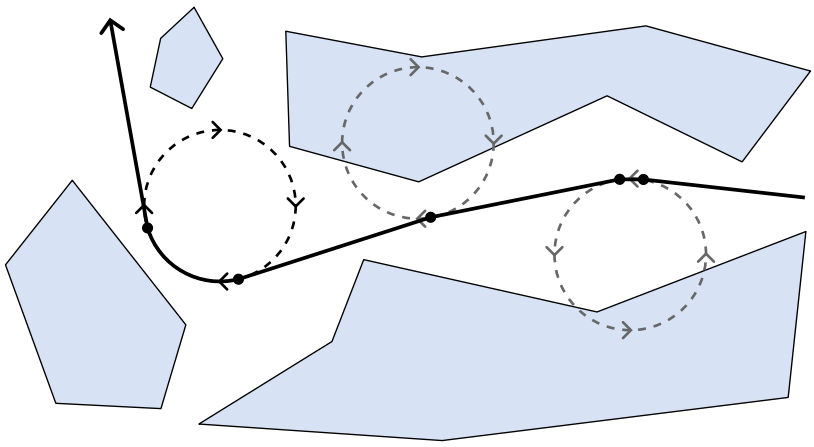


Figure 4.1: Example path containing two elided sharp turns followed by a normal rounded turn. Because elided turns do not use loiter commands, their turn circles are allowed to intersect obstacles. Normal rounded loiter turns must have the entire turn circle in free space.

the target heading. There are certain cases where external disturbances can cause the aircraft to complete a full loiter circle before exiting; therefore, for safety, we must guarantee that the entire loiter circle is free of obstacles, even when we expect to only traverse a small portion of it. This restriction makes it difficult or impossible to plan paths in free areas smaller than the loiter circle diameter. To avoid this problem, a smooth turn is not required if the change in heading between two transits is less than a configurable threshold. This enables the planner to navigate around tight spaces and narrow corridors.

Planning occurs in a four-dimensional space – Cartesian x and y position, altitude and heading. However, PRM milestones are not directly sampled from this space. Instead, loiter turn centers are sampled from the 3D position space, along with a turn direction (i.e., clockwise or counterclockwise). Samples are chosen from within the rectangular bounding box of the map, between user defined minimum and maximum altitudes. If any portion of the disk bounded by the turn circle does not collide with an obstacle, the turn is added to the roadmap. When a loiter turn is used, the entire turn circle must not collide with an obstacle, as described previously. On the other hand, small loiter turns may be elided and represented by two consecutive transits. In this case, only a small portion of the elided turn must not collide with an obstacle. Figure 4.1 provides an example of a path containing two elided turns. When the turn is initially sampled, it does not have any start or end heading defined; therefore, it is opportunistically added to the roadmap if there is any possibility of

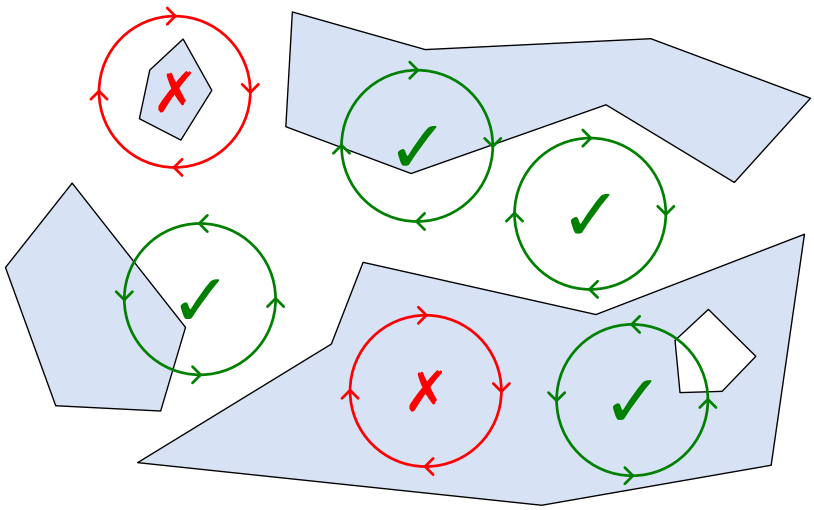


Figure 4.2: Examples of sampled turn circle milestones. The blue filled polygons represent obstacles. Turns in red would be rejected, while those in green would be accepted and added to the roadmap. Note how a turn is accepted if any portion is in free space.

a non-colliding pair of start and end headings. In other words, turn milestones are added if any portion of the disk bounded by the turn circle lies in free space. Collision checking is performed on the disk rather than the turn circle itself because it is far less computationally expensive. This optimization only makes the planner slightly more conservative by not allowing turns around obstacles that are fully contained within the turn circle. Figure 4.2 provides examples of each case.

It is most difficult to find paths within areas that are filled with closely spaced obstacles. Therefore, the planner uses the bridge test to increase sampling density within these regions [37]. A configurable portion of the milestones are sampled with the bridge test, rather than the normal sampling described previously. To sample a milestone using the bridge test, first a pair of locations is sampled from the obstacle space. If the midpoint between the pair of locations does not collide with an obstacle, a turn milestone is added such that a point on the turn circle is tangent to the midpoint. In this manner the line between the two colliding points creates a "bridge" between obstacles. The tangent line is placed perpendicular to the line between the two colliding points, and the turn direction is chosen at random. Two examples of turns sampled with the bridge test are shown in Figure 4.3.

Each sampled turn is then connected to nearby milestones in the roadmap. Connections are attempted to the k-nearest neighbor milestones, where k is a function of the number of milestones n given in Equation 4.4. The derivation of this equation is

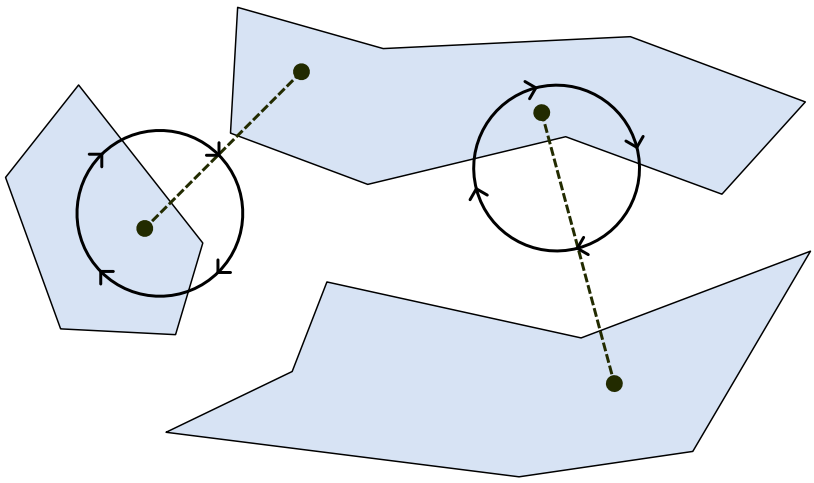
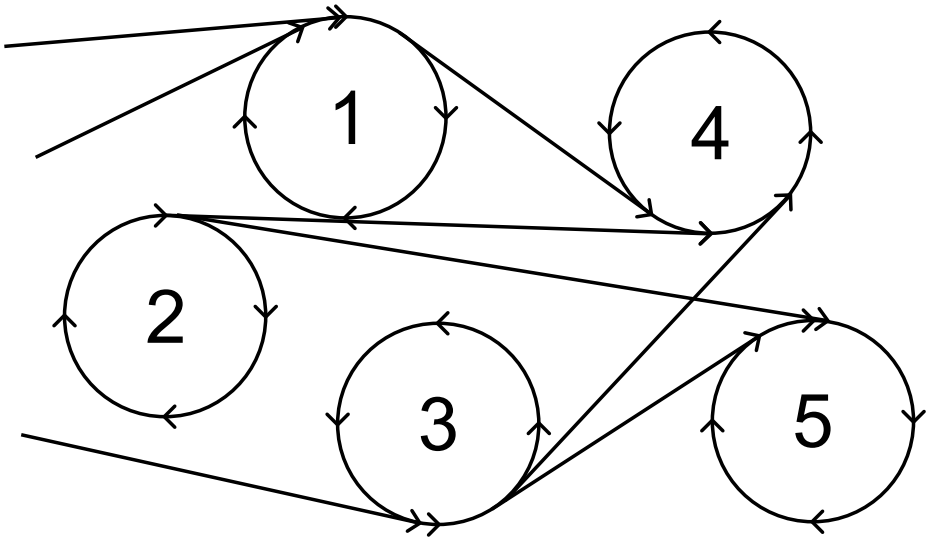


Figure 4.3: Example turn circles sampled with the bridge test. The blue polygons represent obstacles, and a dashed line is drawn between the two sampled colliding points that form the bridge.

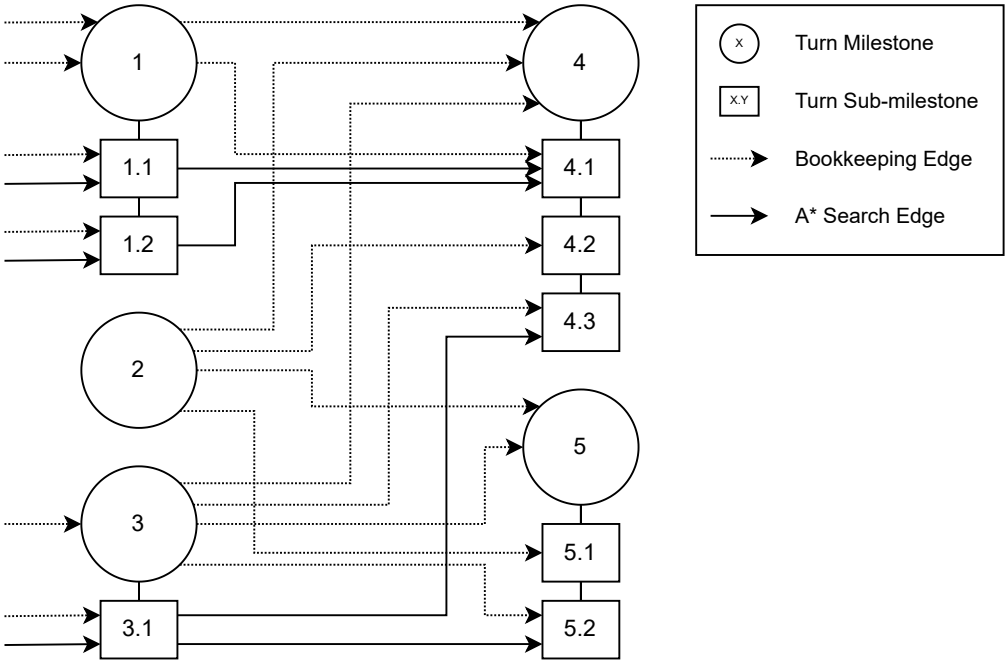
given in [35].

$$k = 2e \log n \tag{4.4}$$

The planner attempts to create a connection to and from each of the nearest neighbor milestones. Each connection results in a directed edge in the graph from the source milestone to the target milestone. The edge has a cost which equals the energy required by the turn around the source milestone plus the energy required by the transit from the source to the target milestone. The source turn energy depends on the incoming heading, which depends on the edge used to arrive at the source milestone. This makes the edge costs path dependent, which is not permissible when searching the roadmap for the lowest cost path using A*. Therefore, an extra step is needed while connecting turn milestones to remove this path dependence. Each time an incoming edge is added to a milestone, a sub-milestone of the new edge's target milestone is created that corresponds to the incoming heading for the transit from the source turn to the target turn. An edge is created between the source parent node and the new target sub-milestone. Additionally, all outgoing edges connecting the target parent milestone to other sub-milestones are replicated on the new target sub-milestone. A* search is performed over the sub-milestones only, eliminating the path dependence. Figure 4.4 illustrates how sub-milestones are connected.



(a) Small subset of milestones from a PRM roadmap, along with the connections between them, shown in the workspace.



(b) Schematic representation of the milestones and connections from Figure 4.4a, showing the sub-milestones that are created to avoid path dependent costs. Only the solid edges are included in the A* search; the others are only used when determining how to connect new milestones to the roadmap.

Figure 4.4: Diagram illustrating how sub-milestones are used to resolve path-dependent costs during A* search.

A connection between milestones is only made if the transit does not collide with obstacles. If any part of the milestone turn circle collides with an obstacle, connections between sub-milestones are only created if the net heading change through the turn is small enough to allow the loiter turn to be elided. Cost calculations and feasibility checks that depend on the wind are not performed until a planning query is made, as the wind conditions may change over the course of the flight.

The planner supports several types of start and goal conditions, each of which is represented in the roadmap by a corresponding type of milestone. Plans may start from a certain state (3D position and heading) in midair, from a loiter circle or from a takeoff state. The supported goals are a 3D position (without a heading constraint), a loiter circle or a landing location. The most commonly used start conditions are a loiter circle or a takeoff from the ground. A loiter provides a stable condition that the aircraft can hold while planning, which allows it to smoothly transition into executing the plan. Internally, both the start and goal loiter conditions are represented similarly to a turn milestone, except that they do not require any complex logic to handle turn elision. No part of the loiter circle is ever allowed to collide with an obstacle. Takeoff milestones are always connected to exactly two turn milestones, both tangent to the end state of the takeoff, but with opposite turn directions. This allows the planner to choose the best direction to turn after taking off. Landings use a similar approach, except there are a range of possible landing headings. The heading range is primarily determined by the wind, while obstacles may additionally make certain headings infeasible. The heading bounds are set by maximum crosswind and tailwind limits based on the aircraft's capabilities. The landing headings are constrained such that the wind component perpendicular to the direction of flight at each heading does not exceed the crosswind limit and the wind component in the direction of flight does not exceed the tailwind limit. If the wind speed is less than both the crosswind and tailwind limits, all headings are allowed. Landing with a tailwind is generally undesirable, but it is useful to allow a small tailwind to improve planning results when the wind speed is too small to obtain a reliable wind direction estimate. The combination of the crosswind and tailwind constraints may result in up to three non-contiguous ranges of allowable headings. Examples of landing heading ranges under different wind conditions are shown in Figure 4.5. Headings within this range are sampled using a base-2 van der Corput low discrepancy sequence. A minimum of 16 samples are taken to allow a reasonable range of possible headings in the case where there are no nearby obstacles. Sampling continues until at least 5 feasible landing

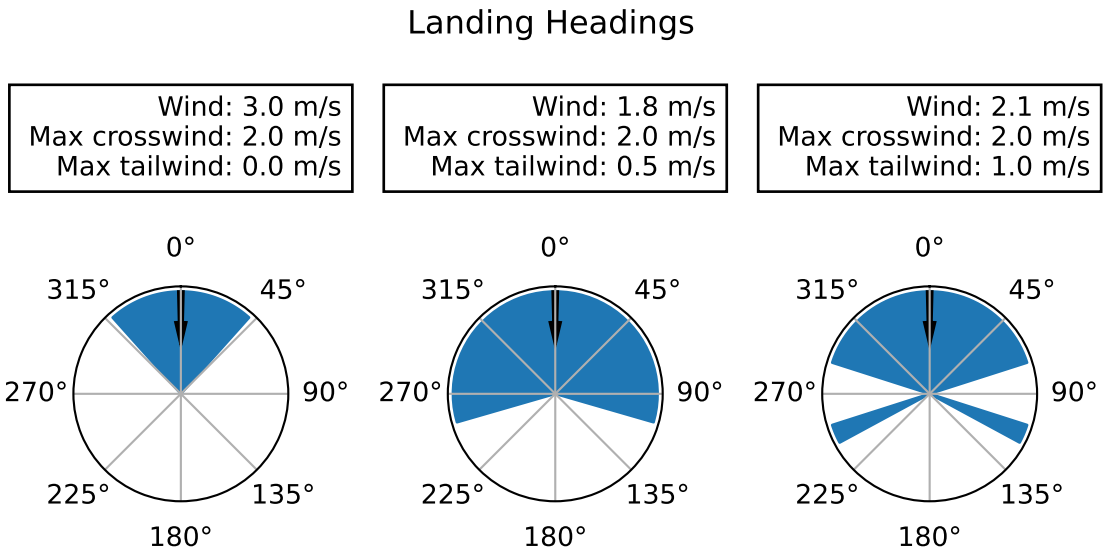


Figure 4.5: Ranges of allowable landing headings under different wind conditions.

headings are found, or the limit of 512 sampling attempts is reached. This adaptive approach allows the planner to use a small discrete step size to find landings around dense obstacles, while not making the A* search slow by adding too many landing options in open areas. Each of the feasible landings is connected to a pair of turn milestones in opposite directions tangent to the start of the landing, allowing the planner to choose both the best landing heading and incoming turn direction.

A* search is used to find the optimal path through the roadmap. The A* heuristic is the energy required to make a straight transit from the source to the target node. Given the complexity of the power model, it is difficult to prove that this heuristic will be admissible in all cases, but it produces reasonable results in practice. As discussed previously, only sub-milestones of turn milestones are included in the search to avoid path dependent costs. The planner incorporates a wind model, which predicts the wind vector at each point in space. The energy cost calculation depends on the wind, and the wind model is also used to ensure the maximum climb or sink rate is not violated. The aircraft climb/sink rate limits with respect to time (i.e., in units of meters/second) do not depend on the wind, but the feasible climb slope (i.e., meters climbed per meter of horizontal distance) is wind dependent. For example, a climb between two points that is feasible with no wind may become infeasible with a tailwind because the time required to cover the horizontal distance decreases. Additionally, the aircraft cannot reach its maximum climb/sink rate instantaneously, which further

limits the average climb or descent altitude, particularly for short transits. The target climb rate is filtered and limited by the flight controller, and the planner models these factors to determine the maximum achievable altitude change between two points. The details of this modeling are described in the energy estimation chapter.

The cost calculation is the most expensive part of the search, so several techniques are used to reduce the number of cost calculations that must be performed. First, all costs are cached, allowing multiple queries under the same wind conditions to be answered efficiently. Secondly, the outgoing transit costs for each milestone are calculated separately and reused for each of the sub-milestones. Each sub-milestone corresponds to a particular turn start heading, and therefore each has a different turn cost. However, each sub-milestone is likely to share many of the same outgoing neighbors, and all sub-milestones will use the same transit to reach a particular neighbor. Therefore, the transit cost only needs to be calculated once per neighbor and then added to the sub-milestone specific turn cost to get the cost for a particular edge.

After the A* star search completes, the initial path is further refined using a shortcutting optimizer. This optimizer repeatedly picks a pair of turns in the path and attempts to generate a direct transit between them. If this transit is feasible and reduces the overall cost of the mission, it replaces the existing segments connecting the two turns. This process is repeated 100 times or until there are no more segments that could possibly be removed. The 100 iteration limit was chosen empirically to significantly exceed the maximum number of useful iterations observed during the experiments.

Chapter 5

Landing Control

The most complex part of fixed-wing flight is landing. Our approach is designed to work using only the IMU and inaccurate barometric altimeter included in standard flight controllers. Compared to other approaches, it minimizes the number of sensors required for landing and achieves safe and controlled landings with minimal tuning. Landing begins with a steep descent at a fixed slope angle until the aircraft reaches the flare altitude (usually 4-6 meters above the ground). During flare, the aircraft pitches up to hold its nose a few degrees above level and begins to use throttle alone to maintain a constant sink rate. This combination of constant pitch and sink rate should produce a constant airspeed. The desired pitch and sink rate must be chosen to ensure the resultant airspeed stays above the stall airspeed. This landing profile is illustrated in Figure 5.1.

This technique differs from other common auto-landing algorithms, such as the one normally implemented in the ArduPilot flight control software. With ArduPilot, the motor is stopped during flare, and pitch is used to control airspeed via feedback from an airspeed sensor. Additionally, there is a minimum pitch limit to keep the aircraft from impacting the nose wheel into the ground or diving the front of the floats underwater. With the nose high and the motor off, the aircraft will eventually run out of kinetic energy and stall, so touchdown must occur quickly. In practice, this means the aircraft would require a rangefinder to trigger the flare at precisely the right distance above the ground. This approach requires two extra sensors over our method: an airspeed sensor and a rangefinder. Airspeed sensors are problematic when operating on water, as the most common type uses differential pressure measurements from a pitot tube, which can easily be fouled by water [38]. In addition, it is difficult to find a low-cost rangefinder that works reliably over water. A sensor with adequate

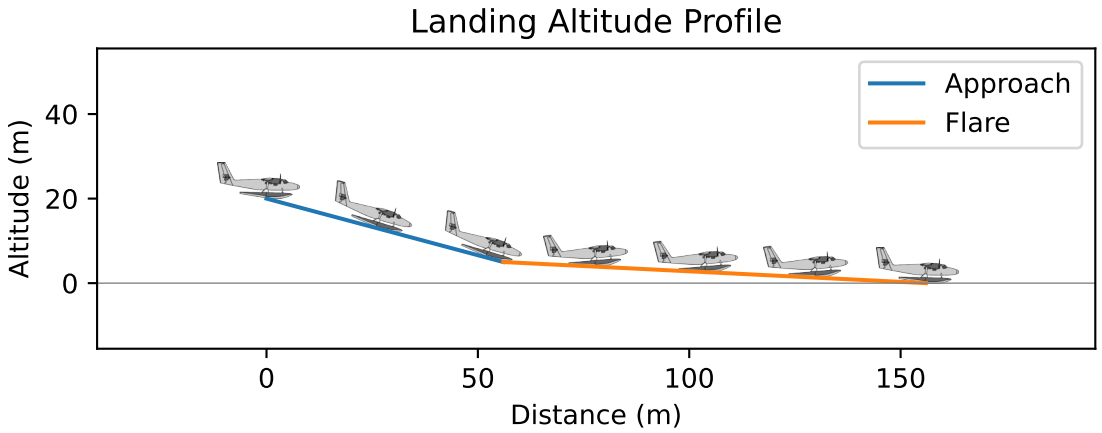


Figure 5.1: Plot of the aircraft's altitude during landing approach and flare. The aircraft's attitude is also shown at various points along the landing.

performance would make up a large percentage of the overall cost of the system.

Our approach avoids both of these sensor requirements. We control airspeed indirectly, using pitch and sink rate, which are easy to measure without specialized sensors. By keeping the motor on, we can continue the flare indefinitely, allowing us to trigger flare at a higher altitude to prevent premature collision with the ground even when using a somewhat inaccurate altitude measurement from a barometric altimeter.

Chapter 6

Energy Estimation

The path planner requires a mission energy estimate to generate energy optimal paths. Additionally, the mission manager requires an energy estimate to ensure the aircraft will be able to visit all landing locations while respecting the energy budget. During normal flight (transits and turn segments), we estimate energy by numerically integrating an estimate of the instantaneous power consumption over the planned trajectory. For takeoff and landing flare, we assume a fixed average power consumption. This is multiplied with the time estimate for the segment to obtain the total energy.

Section 6.1

Power Model

A power model for fixed-wing aircraft can be derived from physical laws, as is done in [33]. The final model of steady state power derived in that paper is as follows:

$$P = \frac{dE}{dt} = K_p \frac{v_a^3}{\eta} + K_i \frac{\cos^2 \gamma}{\eta v_a \cos^2 \phi} + mg \frac{v_a \sin \gamma}{\eta} \tag{6.1}$$

where K_p and K_i are constants that depend on the aircraft's aerodynamic properties, η is the combined motor and propeller efficiency, v_a is the current airspeed, m is the aircraft mass, g is the gravitational acceleration, γ is the climb angle (equal to the pitch angle in steady state flight), and ϕ is the roll angle. There are several difficulties involved in applying this model directly to a planning problem. First, it contains several unknowns that are difficult to determine without extensive aircraft characterization. η depends heavily on both airspeed and motor speed, and likely can

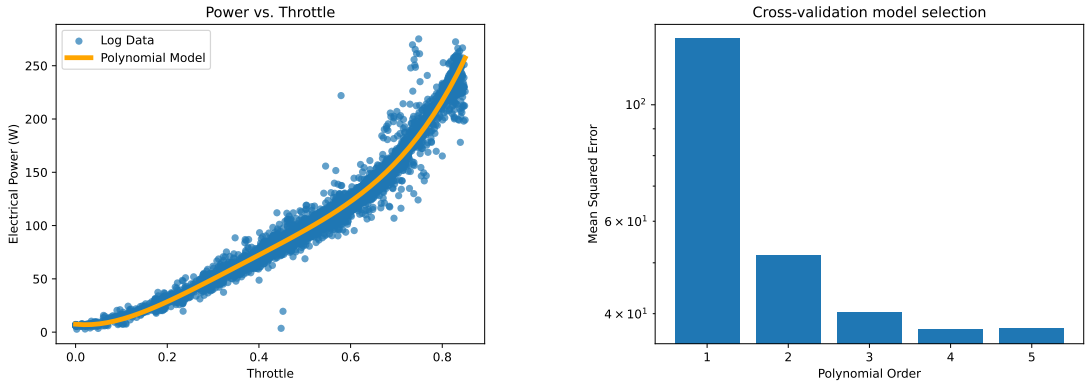
only be accurately modeled using data collected on a thrust stand in a wind tunnel. With an efficiency model, it may be possible to determine K_p and K_i from flight data. However, even if all this information was known, the airspeed is still difficult to predict while planning. Without an airspeed sensor, the flight controller can only perform open loop control of airspeed. Pitch is controlled to maintain the desired climb rate, and throttle is set using a piecewise linear function of pitch. This generally maintains a safe airspeed, but it is difficult to accurately predict the airspeed that will result from a particular climb rate. Additionally, the electrical power is not linear with respect to throttle due to varying efficiency and airspeed.

Given the difficulty in applying the previously described power model to the planning problem, we chose to develop a simpler model derived from experimental data and the control laws used by the flight controller. In the absence of disturbances, the target pitch of the aircraft is set as a function of the climb rate, and the throttle is in turn a function of the target pitch and roll. We then map throttle to power using a fourth-order polynomial model fit to experimental data.

The mapping from climb rate to pitch is described by the piecewise linear relationship given in Equation 6.2. In addition to a feed-forward term from climb rate to pitch, the flight controller includes proportional and integral feedback terms to correct for altitude error. The proportional term should be close to zero on average, while the integral term should settle on a steady state pitch required to maintain constant altitude. With proper configuration of the aircraft cruise attitude, the integral term should remain at zero, but in practice it effectively applies a small constant pitch offset, represented by γ_0.

$$\gamma = \gamma_0 + \begin{cases} \frac{v_c}{v_{c,max}} (\gamma_{max} - \gamma_0) & v_c \geq 0 \\ \frac{v_c}{v_{c,min}} (\gamma_{min} - \gamma_0) & v_c < 0 \end{cases} \tag{6.2}$$

Equation 6.3 describes the relationship between pitch (γ), roll (ϕ) and throttle (T). This consists of a piecewise linear relationship between pitch and throttle and a term that applies additional throttle to compensate for induced drag in turns. The induced drag term gain depends on a flight controller parameter, which we refer to as k_{roll}. When calculating the energy required for a transit, the roll is assumed to be zero, and therefore the induced drag term is also zero. Assuming roll and pitch are known accurately, this equation will exactly reproduce the throttle that will be applied by the flight controller.



(a) Plot of power measurements versus throttle as well as the fourth order polynomial model used to model this relationship.

(b) k-fold cross-validation (k=10) mean squared error for different orders of polynomial model.

Figure 6.1: Data-driven energy model.

$$T = T_{cruise} + k_{roll}\frac{1}{\cos\phi - 1} + \begin{cases} \frac{\gamma}{\gamma_{max}}\left(T_{max} - T_{cruise}\right) & \gamma \geq 0 \\ \frac{\gamma}{\gamma_{min}}\left(T_{min} - T_{cruise}\right) & \gamma < 0 \end{cases} \tag{6.3}$$

After pitch has been determined from climb rate and throttle from pitch, we map throttle to power using a fourth-order polynomial fit to power data derived from voltage and current measurements on board the aircraft. This fit uses robust regression with a Huber loss function. Figure 6.1a shows the experimental data and polynomial model. The polynomial order was selected by minimizing the mean squared error (MSE) using k-fold cross-validation with k=10 to avoid overfitting. Figure 6.1b shows the cross-validation MSE for different polynomial orders. This relationship should be largely independent of battery state of charge because the flight controller automatically scales the throttle output to compensate for changing battery voltage.

Section 6.2

Altitude Model

Combining these equations allows instantaneous power to be determined from the instantaneous climb rate. The target climb rate then depends on the flight controller's altitude controller. The target altitude is filtered using a pair of first-order low pass

filters. The climb rate limit is applied between the first and second filter and the final target climb rate used to compute the target pitch is computed from the differences in filtered target altitude between two time steps. We assume the target climb rate is met exactly, as it is difficult to account for disturbances when performing energy estimation. For small altitude changes (less than 15 meters by default), a step change in target altitude h_{in} is applied to the low-pass filters. For larger altitude changes, h_{in} is interpolated between the start and end altitudes as the aircraft travels between waypoints. This effectively applies a ramp input to the low-pass filter, resulting in a phase delay equal to the filter time constant. These filters are modeled by the system of ordinary differential equations (ODEs) given in Equations 6.4 and 6.5. h_0 represents the output of the first filter, and h_{out} is the final output target altitude, the time derivative of which is the climb rate v_c. τ_1 and τ_2 are the time constants of the first and second filters, respectively.

$$\frac{dh_0}{dt} = \text{clamp}\left(\frac{h_0 - h_{in}}{\tau_1}, v_{c,min}, v_{c,max}\right) \tag{6.4}$$

$$v_c = \frac{dh_{out}}{dt} = \frac{h_{out} - h_0}{\tau_2} \tag{6.5}$$

The phase delay in the altitude controller causes the aircraft to consistently arrive below or above the waypoint at the end of each climb or descent, respectively. If not compensated for, this would result in an error in the start altitude of the next mission segment, making the subsequent predicted path inaccurate. To compensate for the phase lag, the planner always includes another waypoint at the target altitude placed a certain distance before the end position of the transit. This produces a section of flat target altitude that gives the aircraft controller time to level out and asymptotically approach the target altitude. The length of this level out segment is chosen to guarantee that the aircraft will arrive at the end of the transit with less than 2 meters of altitude error. The minimum level out distance required to satisfy this constraint cannot be determined except through iterative optimization, which would be unacceptably slow in this application. Therefore, a heuristic is used to over-estimate the minimum level out distance. If the heuristic produces a distance longer than the total length of the transit, the aircraft is commanded to climb or descend as fast as possible. If this results in a final altitude more than two meters from the target altitude, the transit is rejected as infeasible. This approach guarantees

safety, but often results in transits that are steeper than required, potentially reducing efficiency.

Figure 6.2 shows several transit altitude and climb rate profiles produced by the altitude controller model under different conditions. These include a level out segment where applicable.

Lastly, groundspeed is needed to convert time derivatives such as power and climb rate into distance derivatives. Groundspeed refers to the horizontal speed of the aircraft relative to the ground. It is a function of airspeed, climb rate and wind. As discussed previously, airspeed is difficult to estimate, but for groundspeed calculations we use a piecewise linear function of the climb rate:

$$v_a \approx \begin{cases} v_{a,cruise} & v_c \geq 0 \\ v_{a,cruise} + \frac{v_c}{v_{c,min}}(v_{a,cruise} - v_{a,max}) & v_c < 0 \end{cases} \tag{6.6}$$

For positive climb rates, the function is flat, and the airspeed remains at the level cruise speed $v_{a,cruise}$. For negative climb rates, the airspeed increases linearly until it reaches a maximum value $v_{a,max}$ at the minimum climb rate (i.e., maximum sink rate) $v_{c,min}$.

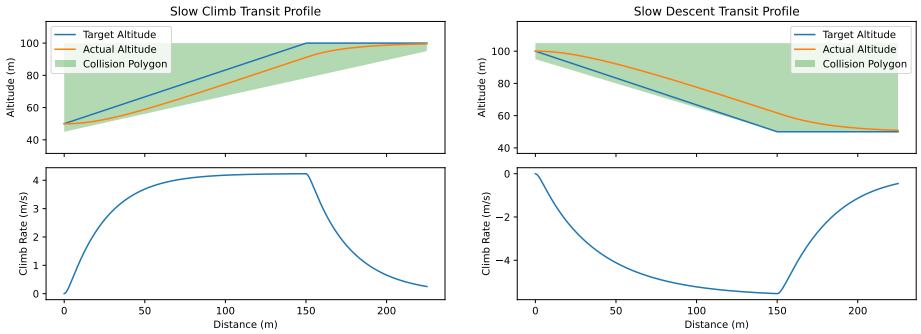
Given an estimate of airspeed, wind and climb rate, the ground speed (v_g) is computed according to Equation 6.9. If the radicand becomes negative, this indicates that the wind conditions make it impossible to maintain the desired course.

$$\mathbf{w} = \begin{bmatrix} w_x & w_y \end{bmatrix} \tag{6.7}$$

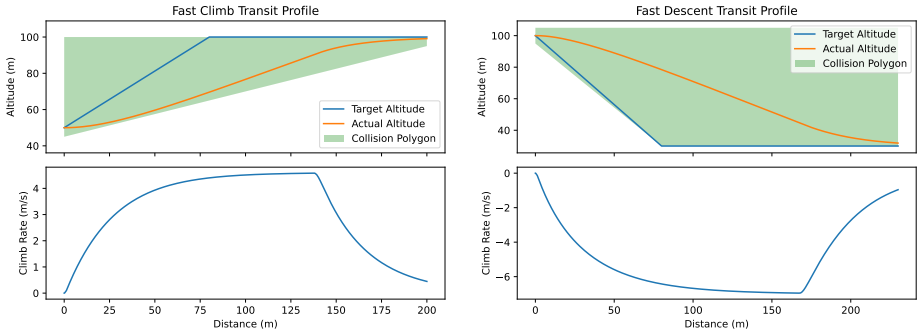
$$\mathbf{d} = \begin{bmatrix} d_x & d_y \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta \end{bmatrix} \tag{6.8}$$

$$v_g = \mathbf{d} \cdot \mathbf{w} + \sqrt{2w_x w_y d_x d_y - w_x^2 d_y^2 - w_y^2 d_x^2 - v_c^2 + v_a^2} \tag{6.9}$$

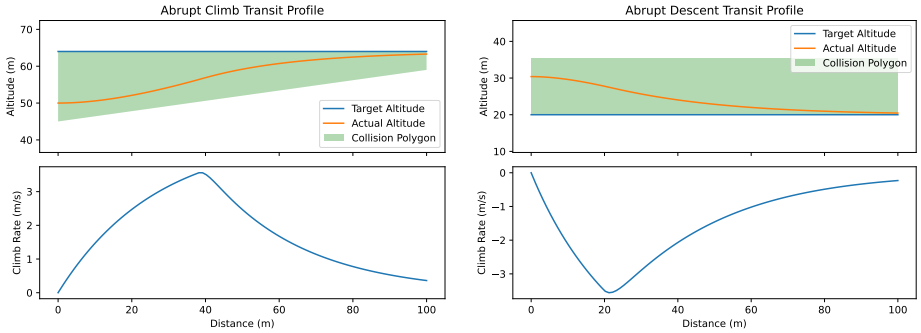
With these relationships defined, we can integrate a system of ODEs to determine the power over the length of the transit segment to determine total energy. Our implementation uses the Dormand–Prince method [39], a popular technique for solving ODEs.



(a) Gradual climb and descent such that the target slope remains below the maximum achievable climb rate. The actual altitude lags behind the target altitude by a fixed amount.



(b) Fast climb and descent such that the target altitude slope exceeds the maximum achievable climb rate. The actual altitude diverges from the target altitude, requiring a longer flat region to allow the altitude to catch up.



(c) For altitude changes below 15 meters, the flight controller does not calculate a slope. Instead, the target altitude is changed abruptly, and the aircraft climbs or descends as fast as it can.

Figure 6.2: Plots of transit climb/descent side profiles. Except for the small altitude change transits in (c), all transits consist of one waypoint that establishes the climb/descent slope, and one waypoint that establishes a flat target altitude segment to allow the aircraft to level out and approach the final desired altitude. The blue target altitude lines represent the input to the vertical rate filter (h_{in}), while the orange lines represent the output of the filter (h_{out}). The green shaded areas represent the polygons that are checked for collisions.

$$\frac{dh_0}{dx} = \frac{dh_0}{dt}\frac{dt}{dx} = \frac{dh_0}{dt}\frac{1}{v_g} \tag{6.10}$$

$$\frac{dh_{out}}{dx} = \frac{dh_{out}}{dt}\frac{dt}{dx} = \frac{v_c}{v_g} \tag{6.11}$$

$$\frac{dE}{dx} = \frac{dE}{dt}\frac{dt}{dx} = \frac{P}{v_g} \tag{6.12}$$

For turns, a similar approach is used, except that the differential equations are in terms of heading and the roll angle used when estimating throttle in Equation 6.3 is no longer zero. The roll angle is given in Equation 6.13, where r is the turn radius and g is the gravitational acceleration.

$$\phi = \frac{v_g^2}{rg} \tag{6.13}$$

$$\frac{dh_0}{d\theta} = \frac{dh_0}{dt}\frac{dt}{d\theta} = \frac{dh_0}{dt}\frac{r}{v_g} \tag{6.14}$$

$$\frac{dh_{out}}{d\theta} = \frac{dh_{out}}{dt}\frac{dt}{d\theta} = \frac{rv_c}{v_g} \tag{6.15}$$

$$\frac{dE}{d\theta} = \frac{dE}{dt}\frac{dt}{d\theta} = \frac{rP}{v_g} \tag{6.16}$$

The aircraft applies approximately constant throttle during takeoff, so the average power is nearly the same in all takeoffs. We then need to estimate the time required for takeoff. We model takeoff as a period of constant acceleration until the aircraft reaches the takeoff airspeed, followed by a period of constant climb rate until the takeoff end altitude is reached. Under this model, the takeoff time is given in Equation 6.18

$$v_{a,start} = -w_x \cos\theta - w_y \sin\theta \tag{6.17}$$

$$t_{takeoff} = \max\left(\frac{v_{a,takeoff} - v_{a,start}}{a_{takeoff}}, 0\right) + \frac{h_{takeoff}}{v_{c,takeoff}} \tag{6.18}$$

As discussed in the path planning chapter, the planner uses a takeoff command to climb only half the desired takeoff altitude, while a transit segment is used to climb

the second half. The model described above is only used for the takeoff command; the transit is modeled in the standard manner.

Landing consists of two phases, approach and flare. Approach is similar to a transit, except the altitude controller compensates for the phase delay caused by the low pass filters. We therefore assume the aircraft exactly follows the line between the start of the approach and the start of the flare. To determine the climb rate, we first calculate the aircraft speed along this line (the "3D" groundspeed $v_{g,3d}$ that includes the vertical component) using Equation 6.22. The climb rate can then be calculated by extracting the vertical component of this groundspeed (Equation 6.23).

$$\mathbf{w} = \begin{bmatrix} w_x & w_y & 0 \end{bmatrix} \tag{6.19}$$

$$\mathbf{d} = \begin{bmatrix} x_1 - x_0, & y_1 - y_0, & h_1 - h_0 \end{bmatrix} \tag{6.20}$$

$$\hat{\mathbf{d}} = \begin{bmatrix} d_x & d_y & d_h \end{bmatrix} = \frac{\mathbf{d}}{|\mathbf{d}|} \tag{6.21}$$

$$v_{g,3d} = \hat{\mathbf{d}} \cdot \mathbf{w} + \sqrt{(v_{a,app}^2 - w_y^2)d_x^2 + (v_{a,app}^2 - w_x^2)d_y^2 + (v_{a,app}^2 - w_x^2 - w_y^2)d_h^2 + 2w_x w_y d_x d_y} \tag{6.22}$$

$$v_c = \text{clamp}\left(d_h v_{g,3d},\, v_{c,min},\, v_{c,max}\right) \tag{6.23}$$

The approach energy can be calculated by integrating Equation 6.24 over the 3D length of the approach segment.

$$\frac{dE}{dx} = \frac{dE}{dt}\frac{dt}{dx} = \frac{P}{v_{g,3d}} \tag{6.24}$$

Finally, we calculate the flare energy by multiplying the average power during flare by the flare time, in Equation 6.25.

$$E = \frac{P_{flare} h_{flare}}{-v_{c,flare}} \tag{6.25}$$

Chapter 7

Wind Estimation

Wind plays a large role in both planning and energy estimation for fixed-wing aircraft. Wind affects takeoff and landing trajectories and is included in the time and energy estimation equations described in the previous chapter. It is possible to estimate the wind vector during flight using a zero sideslip assumption. This approach is implemented in ArduPilot and provides an instantaneous wind estimate at each point during the flight. During planning, one approach is to assume a constant wind field that matches the latest wind estimate, but this assumption does not usually hold under real wind conditions. In particular, the wind speed tends to increase with altitude, and an aircraft that spends most of its time at high altitudes will overestimate the wind at ground level, leading to an inaccurate landing.

Our planner supports arbitrary spatial wind models, where the wind vector can vary continuously over 3D space. The wind model does not allow for changes in wind as a function of time, but this restriction could be easily removed. This means that the model itself currently does not include time as a variable, not that the model cannot change over time (between different planning queries). In practice, we use a linear model of wind speed versus altitude. The linear model is fit to instantaneous wind speed and altitude measurements collected during flight using recursive least squares estimation, while the wind direction is assumed to match the latest measurement. More complex models may be the subject of future work.

Chapter 8

Water Sampling Mission

While there are multiple possible applications of our work, to demonstrate our proposed planner, landing approach, and energy estimator, we consider a water sampling mission. Our demonstration water sampling mission is orchestrated by a mission manager component. This component receives input from the user, runs the path planner, and communicates the results to the flight controller. The sampling mission consists of two phases: a survey phase and a sampling phase. During the survey phase, the aircraft flies a survey pattern while using remote sensing to locate points of interest for direct sampling. The aircraft then visits each of these sampling points during the sampling phase, before finally returning home.

Section 8.1

Survey Phase

The user plans a mission for the survey phase using standard tools such as ArduPilot Mission Planner or QGroundControl. This mission normally contains a takeoff, a lawnmower pattern survey that covers the area of interest, a landing, and finally a marker item to indicate the mission should trigger a sampling mission. If the marker item is missing, the mission manager will ignore the mission. This avoids unintentional activation of the sampling mission logic.

Once the user uploads the mission, the mission manager post-processes the mission and extracts information from it. The manager also looks for a landing command and saves the landing location from this command to use as the return position for the final landing of the sampling phase. Lastly, the manager optimizes the mission to round all sharp corners. Rounding is important because the aircraft has aggressive

navigation tuning to allow it to follow auto-generated missions during the sampling phase with low cross-track error, thereby improving energy estimates and collision detection accuracy. With this aggressive tuning, sharp corners in missions would cause large attitude changes that are normally undesirable.

After validating and processing the mission, the mission manager waits for the user to arm the vehicle and switch to the auto flight mode, which causes the flight controller to start executing the survey mission. As the survey runs, we assume that the aircraft uses remote sensing to search for sampling targets, which are then provided to the mission manager. For this demonstration, we simulate detection of a predetermined set of points. We simulate a camera with a circular field of view. Any sampling point that falls within the circular projection of the camera's cone of vision is added to the list to visit. The simulated camera image plane is always parallel to the water surface; effects of aircraft attitude are not modeled.

While executing the survey phase, the manager periodically uses the path planner to generate plans for the complete sampling phase, starting from the current location, landing at all sampling points and finally landing at the home location extracted from the survey mission.

Section 8.2

Sampling Phase

The order in which the sampling points are visited is determined by solving an asymmetric traveling salesman problem (TSP). The planner is used to calculate the minimum energy cost to travel between each pair of sampling points, as well as the cost to travel from the start location to each sampling point, and from each sampling point to the home location. These costs are arranged into a matrix and the TSP solver provided by Google OR-Tools [40] is used to find the lowest energy ordering of sampling points.

In our scenario, we want to ensure that all discovered sampling points can be visited in a single flight without recharging the battery. The energy estimate for the sampling plan is used to determine the turn-around point where the mission manager should exit the survey phase and begin the sampling phase. As the aircraft flies and discovers more sampling points, the energy remaining in the battery decreases and the energy required to visit all the sampling points increases. Once these two values are equal (with a safety margin), the sampling phase begins. For this to work

correctly, the energy estimate must accurately model the energy consumption of the aircraft. If the estimate were solely used as a cost function for the path planner, it would only need to be accurate up to a scale factor to allow it to be used to compare candidate paths. The requirement for absolute accuracy means that the energy model parameters must be characterized for a specific model of aircraft. This is one of the simplest sampling approaches that serves primarily to demonstrate and validate the path planner and energy estimator.

Once the aircraft reaches the turnaround point, it executes the first planned landing. After landing, the aircraft uses sensors to collect information from the water at that location. After sampling is complete, the aircraft prepares to take-off to visit the next sampling point. Rather than continuing to execute the previous plan, the manager generates a new plan, with the takeoff heading matching the aircraft's current heading. The original plan calculated the takeoff heading to point into the wind, according to its estimate of the wind, which may have some error and may not reflect changing conditions. Once on the ground, the aircraft acts like a weathervane, providing an estimate of the current wind direction. In addition, forcing the takeoff heading to match the current heading avoids any need to attempt to rotate the aircraft. The aircraft then begins to travel toward the next sampling point and repeats the process until the final landing location (from the original survey mission) is reached.

The mission manager is designed to fail safe in case of an error and to avoid ever taking control away from a human pilot that is attempting to override its control. If an error occurs while the vehicle is armed, the manager will send no more commands to the flight controller and will remain in a fail-safe state until the vehicle is disarmed, at which point it will restart. Inaction by the mission manager should never result in an aircraft safety issue, even if the human pilot does not immediately take over. If an error occurs while the vehicle is disarmed, the manager delays before restarting, to avoid getting stuck in a fast restart loop. If the human pilot changes flight modes while the manager is in control, the manager goes into an idle state. This avoids the possibility of the manager fighting with the human pilot for control of the aircraft. As soon as the flight mode returns to what the manager expects, it resumes the mission from the last saved state transition. For example, if the survey phase is interrupted, returning to auto mode will simply continue the survey from where it left off. If a landing mission is interrupted, resuming will cause a new mission to be planned from the current position and automatically executed.

In addition to the primary sampling mission workflow, the mission manager sup-

ports planning and executing one-off missions. The user provides an outline of the mission, potentially including waypoints, loiters, and a landing location. After the user uploads the outline, the planner generates a full mission that takes into account wind conditions and obstacles. This mode is useful for demonstrating planner performance in difficult landing scenarios, particularly on land.

Chapter 9

Experiments

Our approach is designed to work with a wide range of aircraft, using only standard sensors found on off-the-shelf flight controllers. We ran our experiments on a low-cost platform based on the E-Flite Twin Otter model airplane with a Matek H743-WING V2 flight controller running the open source ArduPilot firmware. This flight controller fuses measurements from a 9-axis inertial measurement unit (IMU), barometric altimeter and GPS and executes the attitude and navigation control loops. A Raspberry Pi 4 Model B with 2 GiB of RAM was included onboard to run the planning algorithm, communicating with the flight controller using the MAVLink protocol over UART. The total cost of all components was approximately US$600. This aircraft can be seen in Figure 1.1. The path planner and energy estimator were implemented in Rust. The Twin Otter worked well for this application because it supports both wheeled landing gear and floats for operation on land and water. Additionally, its dual motors allow differential thrust to improve maneuverability on the water. A flying boat style airframe with a buoyant hull rather than floats would have yielded greater efficiency due to reduced drag, but floats offered greater flexibility while testing.

We ran experiments to test the planner by itself and demonstrate the full sampling mission. We tested the planner on land in different environments, including an open field and a former golf course (see Figure 10.1). The golf course contained rows of trees with areas of flat grass in between. The trees were all less than 40 meters tall, and the gaps between rows of trees were 30-40 meters in most places. This environment provided plenty of obstacles, while still having room to allow the aircraft to takeoff and land. We completed many successful flights in the golf course and other environments while refining the planner.

We ran the sampling mission on three days in Herrick Cove on Lake Sunapee in New London, NH. The first and third days had relatively low wind speeds, while the second day had relatively high wind. We executed a lawnmower pattern survey at an altitude of 70 meters that covered the entire cove and a small portion of the main body of the lake. Simulated sampling points were scattered inside and just outside the cove. At 70 meters altitude, the wind on the first day was from the northeast with speeds between 2-4 m/s. The second day had wind from the northwest between 6-9 m/s, while the third day had wind from the west-northwest with speeds between 3-5 m/s. On the third day, the air was nearly calm at ground level. The same survey was run five times across the three days, although log data was partially lost on one run due to water ingress, and the final run was aborted part way through the sampling phase due to sunset.

In simulation, we compared the behavior of our planner using our proposed energy cost function against a configuration that used a traditional distance cost function. The distance cost function was computed as the sum of the arc lengths of each turn and the 3D straight line distances between the start and end point of each transit. The exact trajectory of the aircraft as determined by the vertical dynamics model was not used in the transit distance calculation. We looked at a realistic scenario on Lake Sunapee where the aircraft takes off from one location and lands at another, with a peninsula serving as an obstacle in between these locations. The aircraft has the option of flying over the peninsula or around it. We generated a large number of plans with different random number generator seeds to account for the probabilistic variation in paths due to different sets of sampled milestones.

Chapter 10

Results

We completed ten autonomous takeoffs and landings on the same day in the golf course. The planner and aircraft performed well in all cases and never collided with obstacles despite the limited open space in the environment. In one case, the aircraft flipped over on landing because the landing point was mistakenly placed in an area of tall grass. An example path flown in the golf course can be seen in Figure 10.1.

During the lake experiments, all landings (24 in total) were completed successfully with no aborts. The energy-aware TSP chose a landing order that was clearly different from what would have been chosen by a simple TSP based on the distance between sampling points. The TSP chose to visit the points in several passes. The distance between most of the points is smaller than the distance required to takeoff and land

Figure 10.1: Aircraft path during a flight between two points in the golf course.

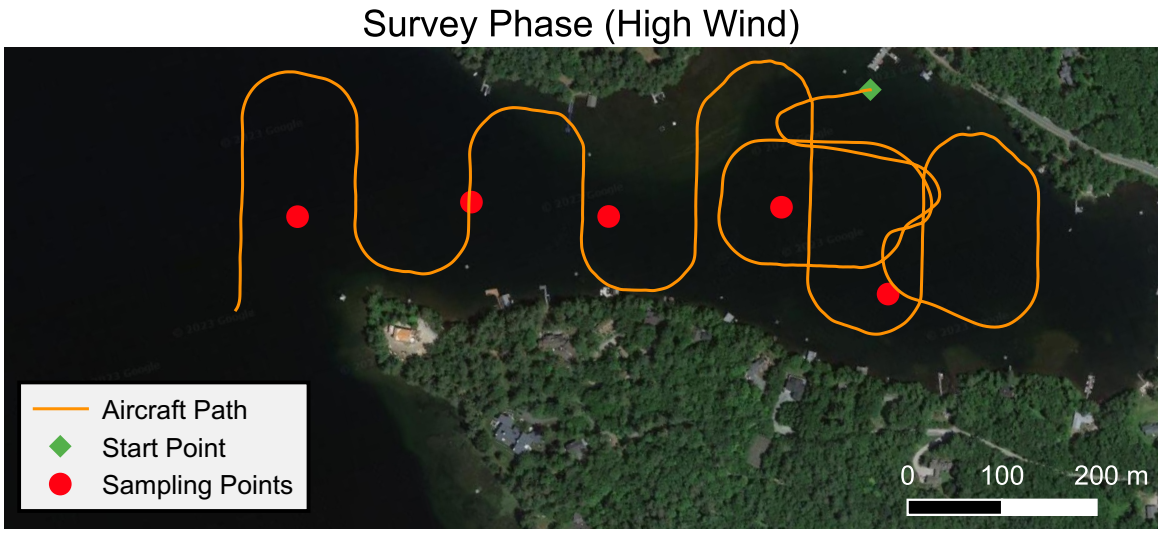
again, so it is more efficient to do multiple passes to avoid having to repeatedly backtrack after each takeoff. This effect can be observed in the ordering of the sampling points in Figures 10.2b and 10.3b. In all but one case, the aircraft reached the energy budget turnaround point before the entire survey was completed. In the final run, decreasing wind speeds allowed the aircraft to complete the entire survey before hitting the energy threshold.

On the day with the highest wind speeds, the aircraft tended to undershoot the landing. The average measured groundspeed was lower than the predicted groundspeed, indicating the wind speed was higher than predicted by the wind model. Additionally, wind shear in the last few meters above the water caused the airspeed to drop, increasing the sink rate. The landing flare controller responded slowly to this disturbance, causing the aircraft to touch down early at a higher sink rate than expected. These issues could be mitigated by an improved wind speed model and more robust control of sink rate in the face of disturbances. The high wind speed also prevented the aircraft from completing enough of the survey to detect all the sampling points. The third day had lower wind speeds, and in this case the aircraft tended to overshoot the landing. This appeared to be attributable primarily to wind estimation and barometer altitude error.

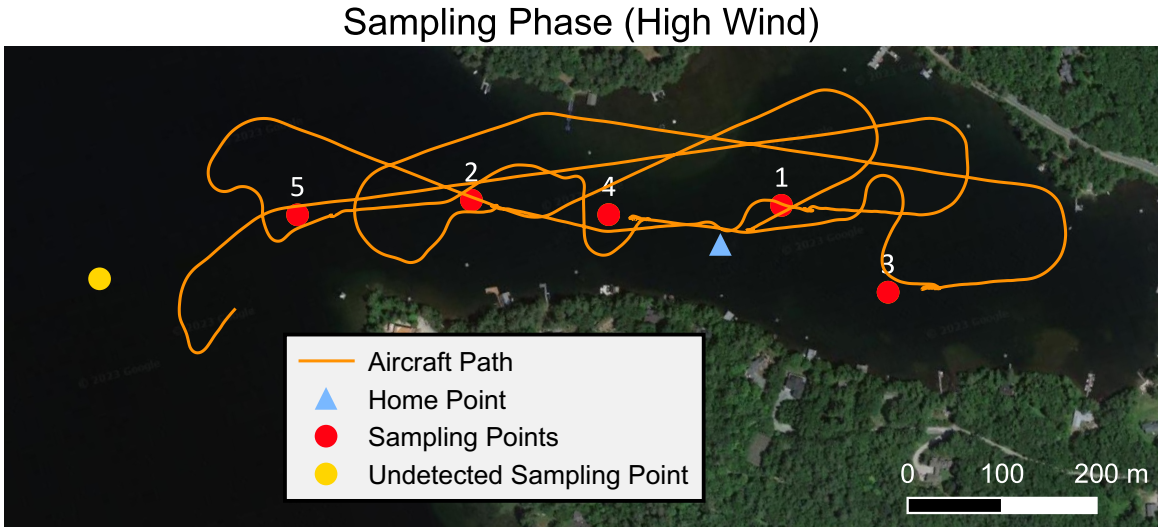
We compared the predicted and actual energy consumption during the experiments. The overall prediction error across all experiments was -5%. The full breakdown of error by command type is shown in Table 10.1. This table shows both the overall energy error for each command type as well as the error weighted by the fraction of total energy consumed by the command type. This latter metric shows how much each command type contributes to the overall error.

The results of our simulated comparison between plans generated using the energy cost function versus the distance cost function varied depending on the chosen wind conditions. If a constant wind field is simulated, with no wind speed dependence on altitude, there is little difference between the plans generated when using each cost function. In this case, optimizing for energy reduces the predicted energy consumption by only approximately 0.4%. With either cost function, the aircraft usually takes the straight route and flies over the obstacle. The aircraft incurs a small energy and distance penalty while climbing to the height required to clear the obstacle, but this is still less than the cost required to take the indirect route around the obstacle.

As discussed earlier, assuming constant wind does not provide a good model of real conditions. If we include a positive linear dependence of wind speed on altitude,

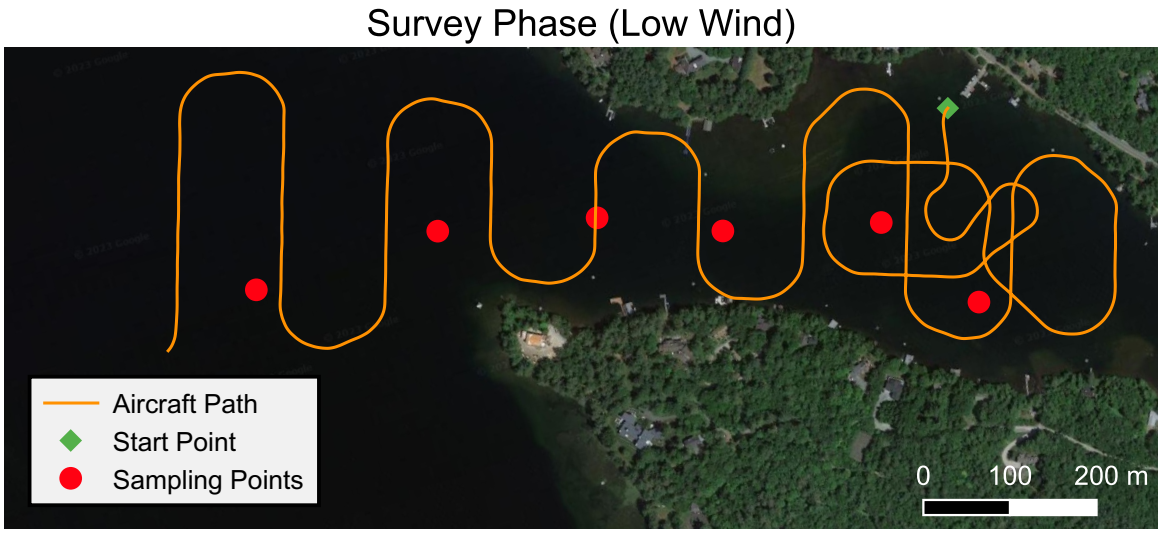


(a) Aircraft path during the survey phase. The aircraft flies a square path at low altitude to improve the wind estimate before starting the lawnmower pattern. The high wind speed slowed the aircraft and prevented the survey from progressing far enough to detect the last sampling point.

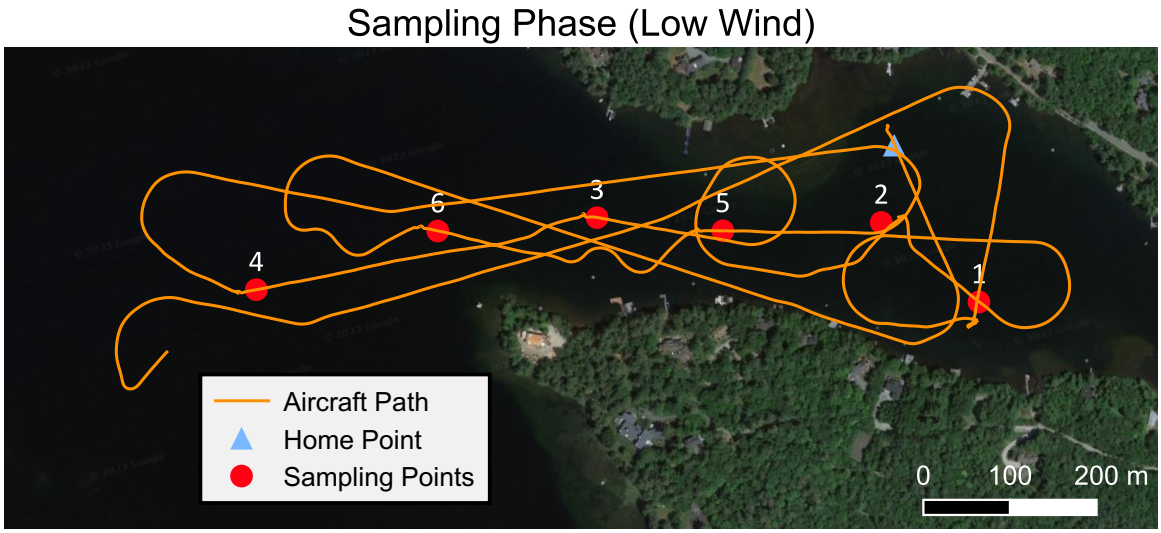


(b) Aircraft path during the sampling phase. The sampling points are numbered in the order they were visited. The aircraft tended to slightly undershoot the target landing point because the linear wind model underestimated the wind speed near the ground and wind shear caused a higher than desired sink rate.

Figure 10.2: Results of a water sampling mission on Lake Sunapee. The wind speed was 6-9 m/s at 70 meters altitude during this run, which is considered relatively high for our aircraft.



(a) Aircraft path during the survey phase. The aircraft flies a square path at low altitude to improve the wind estimate before starting the lawnmower pattern.



(b) Aircraft path during the sampling phase. The sampling points are numbered in the order they were visited. The aircraft tended to slightly overshoot the target landing point because the linear wind model overestimated the wind speed near the ground.

Figure 10.3: Results of a water sampling mission on Lake Sunapee. The wind speed was 3-5 m/s at 70 meters altitude during this run, with little wind at ground level.

Command	Energy Error		Weighted Energy Error
Waypoint	-6.2%	±15.6%	-2.3%
Waypoint (excluding takeoff)	-2.7%	±10.9%	-0.8%
Loiter	15.3%	±45.2%	4.1%
Loiter (excluding start)	0.3%	±19.9%	0.1%
Takeoff	-11.6%	±10.8%	-2.3%
Landing	-18.3%	±15.1%	-2.3%
All	-4.9%	±31.0%	

Table 10.1: Energy estimator error for each type of mission command during the lake experiments. The energy error column gives the estimation error as a fraction of the total energy consumption for the corresponding command type. The energy error uncertainty is given as the standard deviation weighted by the energy consumed by each command sample. The weighted energy error column gives the estimation error as a fraction of the total energy consumed by all commands.

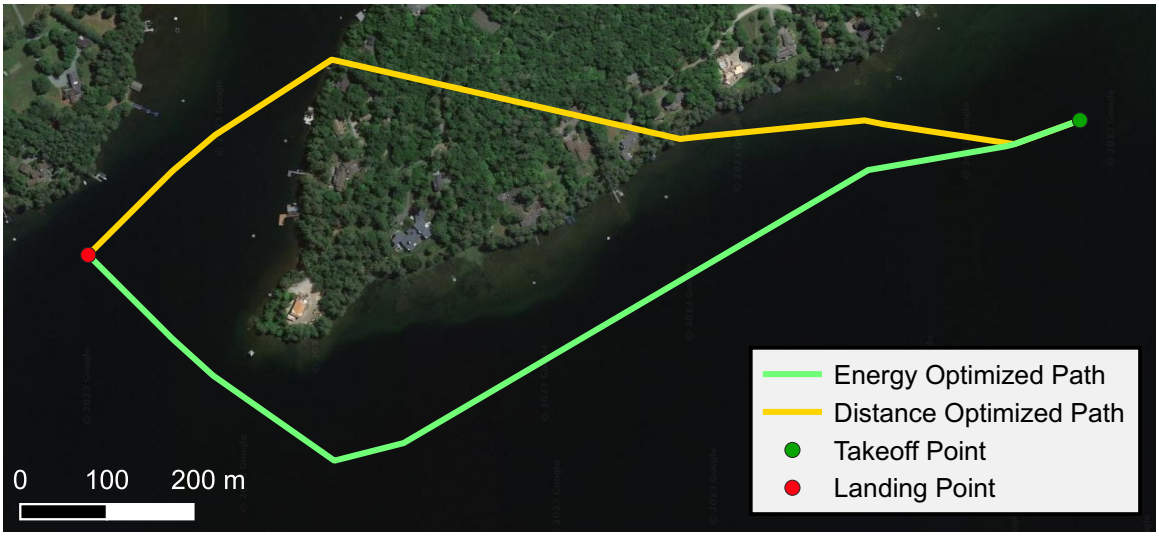


Figure 10.4: Example paths generated in the Lake Sunapee peninsula environment using the energy and distance cost functions. In this scenario, the wind speed increases with altitude. The energy optimized path goes around the peninsula so it can remain at low altitude.

the results change dramatically. If the landing point is upwind of the takeoff point, climbing to a higher altitude incurs a large energy penalty because the aircraft must fight the increased wind. Therefore, flying over an obstacle requires much more energy than remaining at a lower altitude and flying around the obstacle, even though going over the obstacle results in a shorter travel distance. In this simulated scenario, the planner finds paths that use 17% less energy on average when optimizing for energy rather than distance. Example results from this scenario are shown in Figure 10.4.

Chapter 11

Discussion

The experiments showed that our proposed approach allows successful execution of water sampling missions. Here we discuss the main insights we gained through this research and how they will guide our future work.

To operate in confined environments such as the golf course, we overcame several challenges. The first was ensuring that the PRM roadmap had adequate sampling density to allow solutions to be found reliably. During takeoff and landing the aircraft must climb or descend, respectively, near its maximum limit to clear the trees. This requires milestones to be found along the takeoff/landing path at a small range of altitudes. Too low, and the path will not clear obstacles; too high, and the path is not feasible. This problem motivated incorporating the bridge test, which causes more milestones to be sampled in narrow passages, thereby increasing the probability of choosing milestones with the appropriate altitudes. The bridge test largely solved the problem, but sometimes the start and end points needed to be adjusted to successfully generate a plan. In some cases, there may have been no feasible path even if the planner was perfect, but it is difficult to distinguish this scenario from inadequate milestone density or connectivity. The planner generally performed better with a headwind, as this increases the maximum climb/descent slope.

The second issue involved yaw control during takeoff. The flight controller does not perform closed loop yaw control during the first phase of takeoff, so the aircraft tends to turn during takeoff. These turns appeared to be primarily caused by crosswinds, as well as a mechanical bias that caused the aircraft to tend to turn to the left during takeoff. We were unable to conclusively identify the cause of this bias, and it was not detectable at higher airspeeds during normal flight. This behavior can be observed in Figure 10.3b and results in the curving paths seen in some of the takeoffs. On the

open environment of the lake, the lack of yaw control was harmless, but it was more problematic in the golf course where it caused the aircraft to fly closer to obstacles than desired during some of the flights.

We found that the aircraft exhibited less weathervaning behavior than expected. Waves appeared to have more impact on the aircraft heading than wind and tended to turn the aircraft broadside to the waves, placing it perpendicular to the wind. Under the wind conditions during the experiments, this tended to point the aircraft toward land, sometimes preventing the aircraft from having enough room to takeoff again. In these cases, we had to use the radio control transmitter to manually rotate the aircraft to a more favorable heading using differential thrust before resuming the autonomous mission. We experimented with autonomous heading control using differential thrust, but the non-linear interaction between the aircraft and the water makes control difficult, and more work would be required to get satisfactory performance. In fact, it is even challenging to control the aircraft heading with manual control.

Lastly, we ran into difficulty getting the aircraft to precisely land at the desired location. This was primarily caused by inaccurate wind estimates near the ground. The wind speed tends to decrease quickly near the ground, increasing the ground speed and the flare distance. Landing precision would likely be improved by alternative wind models. Additionally, we had issues with barometer altitude measurements. The air flow around the aircraft during flight appeared to cause the measured altitude to be approximately one meter lower than reality. The aircraft travels at approximately 10 m/s during flare with a sink rate of 0.5 m/s, so each meter of altitude error contributes to 20 meters of landing point error. We partially mitigated this issue by including a configurable offset in the planner's predicted flare altitude. Fully solving this problem would require a more accurate source of altitude information, such as a rangefinder. The planner currently assumes the ground is perfectly flat and that landing touchdown will occur at zero altitude relative to the aircraft's home position. This assumption was violated slightly in the golf course, as there was a small altitude difference between some of the takeoff and landing locations chosen for our experiments. This contributed to landing undershoot or overshoot, depending on if the landing point was above or below the takeoff point.

Examining the energy estimation results (Table 10.1) shows that energy estimation error varies between mission segment types. Landings have the largest relative error magnitude. This comes primarily from underestimation of the required flare

time, due to factors discussed previously. Takeoff error comes primarily from errors in determining the takeoff power ($P_{takeoff}$) parameter. In earlier flights that were used to characterize the model, the average takeoff power was lower than during the lake tests. The average waypoint error was biased by waypoint commands that were executed immediately after takeoff. As discussed previously, takeoffs had poor heading control, so the next waypoint command often resulted in a large course correction, increasing estimation error. Excluding these takeoff waypoints greatly reduces the energy estimation error. Loiter commands have a similar bias due to missions that start with a loiter in midair. In these cases, the initial aircraft heading is unknown, so the energy cannot be estimated accurately. We always assume a full circle turn was executed, which tends to result in an overestimate. If these initial loiter commands are excluded, the average error decreases by two orders of magnitude. In reality, the overestimation for initial loiters is not as large as it appears because the aircraft will usually have spent some time in loiter mode while planning before starting the auto mission, but this time is not included in the data.

Planning multiple landings in a water sampling mission requires multiple planning queries under the same wind conditions. This motivated the use of the multi-query PRM* planner, but this choice ultimately presented several unexpected challenges, particularly in achieving adequate performance to enable onboard planning. The difficulty in implementing an efficient PRM* planner stems from the fact that the wind estimate may change over the course of the flight. This limits how much can be computed while building the roadmap. Cost estimates, trajectory feasibility checks (i.e., checking climb rate limits) and even collision detection depend on the wind. In the case of collision detection, an initial permissive wind-independent collision check can be done when building the roadmap, which prevents most infeasible edges from being added. On the other hand, cost calculations and trajectory feasibility checks have to be entirely deferred to query time. This does not entirely eliminate the benefit of a multi-query planner, because generating the water sampling mission requires running several planning queries under the same wind conditions. Therefore, trajectory feasibility and costs can be cached between these queries.

Additionally, our approach is slow to detect that a planning solution cannot be found. Since we do not know for certain which edges are feasible while building the roadmap, we cannot keep track of the connected components of the graph. Therefore, the only way to know that the start and goal are disconnected is to perform an exhaustive search. In practice, connectivity gaps most often occur within a few steps

of the start or the goal. The search begins from the start, so gaps near the start are found quickly, while discovering gaps near the goal requires searching nearly the entire graph.

Our linear wind model is designed to roughly account for the commonly observed positive correlation between altitude and wind speed. Least squares estimation of the linear function is the simplest way to capture this relationship, but there are likely other approaches that would result in better prediction performance. In our experiments, accurate wind modeling is hampered by low quality data provided by the flight controller's side-slip based wind estimator. In particular, the wind estimate is only updated when the aircraft is in a certain range of attitudes. Therefore, the wind estimate is often outdated and does not immediately account for changing conditions as the aircraft climbs or descends. Internally, the flight controller uses an extended Kalman filter to generate the wind estimate, which should provide an uncertainty metric, but this is not currently available externally. If uncertainty information were available, it could be used to weight the inputs to the wind model, or to reject samples with too high uncertainty.

We use a simple recursive least squares estimator to determine our wind model, and this approach means that each new wind measurement has less and less impact on the model coefficients. This causes two problems; first, the model is unable to respond to changing wind conditions. This may become more important if the aircraft flies longer missions. This limitation could be overcome by using a forgetting technique to allow the least squares estimator to follow changes in parameters. The aircraft will spend long periods at constant altitude, so any forgetting solution must not simply consider the age of the measurements, but also whether old measurements are obsolete and can be adequately replaced with new measurements.

In the future, we may explore alternative wind models described in existing climatology literature. Of particular note is the log wind profile, which is a standard model of wind speed in the atmospheric surface layer under certain conditions. This model predicts the wind speed as a function of the natural logarithm of altitude [41, p. 56].

During the mission, the planner runs periodically to check whether the energy required to visit all sampling points indicates that the aircraft needs to end the survey phase and start the landings. This requires computing the full cost matrix for the TSP solver, which can require significant computation time and is quadratic in the number of sampling points. In our experiments, this computation required on the

order of 30 seconds with six sampling points. By the time the planner has finished, the energy estimate is outdated. In the worst case, the turnaround point may occur immediately after starting a planner run, in which case it will not be detected until the following planning run completes. During this time the aircraft will have used around 5% of its total available energy, and, in the absence of a safety margin, may no longer have enough energy to complete all the landings. In practice, we include a safety margin in the energy budget, which somewhat mitigates these problems at the expense of the maximum mission length. Once the sampling phase begins, the aircraft tries to visit all sampling points regardless of the energy budget. This ignores disturbances and energy estimation errors that become apparent as the aircraft flies and, if the safety margin was insufficient, could result in the aircraft running out of energy before it visits all the sampling points.

Chapter 12

Future Work

There are several opportunities to extend and improve our proposed approach. We focused heavily on designing an energy estimator that allows accurate absolute predictions of the energy consumed during the mission. Our approach requires numerically integrating systems of differential equations. It is computationally expensive to integrate large systems of equations while achieving small errors. In the future, we may incorporate additional elements into the cost function to further reduce estimation error, and these are likely to make computation slower. On the other hand, we only require accurate absolute energy estimation when deciding whether a mission satisfies the available energy budget. While running the path planner, only a relative energy metric is needed to decide which path is the most efficient. Additionally, we can tolerate larger energy errors, as small errors during planning do not have a direct impact on the safety or feasibility of the mission and may only result in a slightly sub-optimal path. This motivates the development of a simplified energy metric that neglects some terms of the full model or simply uses a larger integration step size. We can explore the tradeoff between computation time and estimator accuracy. In some cases, such as when the aircraft is loitering while waiting for the planner, a sub-optimal path found quickly may reduce the overall energy consumption.

We demonstrated the efficiency advantages of our energy-aware planner over a traditional distance based planner in simulation but have not yet confirmed these results through real world experiments. We plan to conduct field trials where the aircraft must choose between flying over or around an obstacle under real world wind conditions.

We implemented PRM* to improve the efficiency of multi-query planning, but it is not clear that this benefit is born out in practice. We may want to explore imple-

menting a similar planner using a single-query algorithm, such as rapidly-exploring random trees (RRT), to determine whether a single-query planner may be more efficient in practice. In addition, single-query planners enable better techniques to find paths through difficult environments, which may allow a single-query planner to succeed in cases where PRM* would fail. We have implemented a proof-of-concept planner based on the bidirectional RRT-Connect algorithm [42]. This algorithm is not asymptotically optimal, but initial informal experiments show that it can find a solution to difficult planning problems, such as landing in the golf course, faster than our current planner without a dramatic hit to energy efficiency.

In the future, we would like to relax the assumption that the ground is perfectly level by incorporating a digital elevation model (DEM). The planner could use the DEM to predict the altitude at the landing location as well as to include terrain as an obstacle. This would allow the planner to operate more effectively on land.

In our experiments, detection of sampling points and the sampling process itself was entirely simulated. We would like to develop more realistic approaches to detect areas of interest to sample. As a first step, we could detect easily recognizable targets on the water with a camera. Next, we could apply algorithms for HAB detection to allow the aircraft to find real sampling points. Lastly, we could add a fluoroscope to the aircraft to carry out practical HAB tracking missions.

As discussed previously, the airframe we used for experiments is relatively inefficient, with flight times of approximately 20 minutes, which is not much different from VTOL aircraft such as quadcopters. Fixed-wing aircraft at this scale can be easily capable of a flight time of 40 minutes to an hour, with the most efficient aircraft achieving even longer flights. We would like to explore alternative off-the-shelf and custom designs that can carry larger batteries and have reduced drag. In particular, flying-boat style airframes would enable longer flight times due to the lack of drag from floats.

To demonstrate our planner, we implemented a sampling mission where the aircraft lands at a set of sampling points. Our approach is basic and lacks features that might be desired in practical applications. As discussed previously, the long planning time causes a delay in deciding when the aircraft needs to end the survey phase and start the sampling phase, requiring a large energy budget safety margin. Even without increasing planner performance, this delay could be mitigated by placing the planning start point at a future point in the mission, rather than at the current position. We may also consider tradeoffs between energy allocation to the survey and

sampling phases. There may be cases where it is beneficial to skip a sampling point to spend more time searching for points in the survey phase. We may also explore splitting sampling missions over multiple flights.

Chapter 13

Conclusions

Our proposed energy-aware planner allows energy-efficient and safe combined survey and sampling missions with a fixed-wing UAV. In addition, we show that our simplified landing controller is reliable under real-world conditions without using specialized sensors. The experimental results also show that considering energy and wind is important and leads to significant changes to the order of landings at the identified sampling points.

Overall, our proposed system for fixed-wing UAVs has the potential to be used for environmental monitoring missions that require both surveying and sampling, contributing to the understanding of our planet.

Bibliography

[1] D. Wu, R. Li, F. Zhang, and J. Liu, "A review on drone-based harmful algae blooms monitoring," *Environmental Monitoring and Assessment*, vol. 191, no. 4, p. 211, Apr. 2019, ISSN: 0167-6369, 1573-2959. DOI: 10.1007/s10661-019-7365-8. [Online]. Available: http://link.springer.com/10.1007/s10661-019-7365-8 (visited on 11/04/2021).

[2] C. Kislik, I. Dronova, and M. Kelly, "UAVs in support of algal bloom research: A review of current applications and future opportunities," *Drones*, vol. 2, no. 4, p. 35, Dec. 2018. DOI: 10.3390/drones2040035. [Online]. Available: https://www.mdpi.com/2504-446X/2/4/35 (visited on 11/05/2021).

[3] M. Jeong, M. Roznere, S. Lensgraf, A. Sniffen, D. Balkcom, and A. Q. Li, "Catabot: Autonomous surface vehicle with an optimized design for environmental monitoring," in *Global Oceans 2020: Singapore – U.S. Gulf Coast*, ISSN: 0197-7385, Oct. 2020, pp. 1–9. DOI: 10.1109/IEEECONF38699.2020.9389391.

[4] A. Terada, Y. Morita, T. Hashimoto, *et al.*, "Water sampling using a drone at yugama crater lake, kusatsu-shirane volcano, japan," *Earth, Planets and Space*, vol. 70, no. 1, p. 64, Apr. 19, 2018, ISSN: 1880-5981. DOI: 10.1186/s40623-018-0835-3. [Online]. Available: https://doi.org/10.1186/s40623-018-0835-3 (visited on 04/10/2023).

[5] J.-P. Ore, S. Elbaum, A. Burgin, B. Zhao, and C. Detweiler, "Autonomous aerial water sampling," in *Field and Service Robotics: Results of the 9th International Conference*, ser. Springer Tracts in Advanced Robotics, L. Mejias, P. Corke, and J. Roberts, Eds., Cham: Springer International Publishing, 2015, pp. 137–151, ISBN: 978-3-319-07488-7. [Online]. Available: https://doi.org/10.1007/978-3-319-07488-7_10 (visited on 11/07/2021).

[6] T. Templin, D. Popielarczyk, and R. Kosecki, "Application of low-cost fixed-wing UAV for inland lakes shoreline investigation," *Pure and Applied Geophysics*, vol. 175, no. 9, pp. 3263–3283, Sep. 1, 2018, ISSN: 1420-9136. DOI: 10.1007/s00024-017-1707-7. [Online]. Available: https://doi.org/10.1007/s00024-017-1707-7 (visited on 11/07/2021).

[7] A. S. Barruffa, Á. Pardo, R. Faggian, and V. Sposito, "Monitoring cyanobacterial harmful algal blooms by unmanned aerial vehicles in aquatic ecosystems," *Environmental Science: Water Research & Technology*, vol. 7, no. 3, pp. 573–583, 2021. DOI: 10.1039/D0EW00830C. [Online]. Available: https://pubs.rsc.org/en/content/articlelanding/2021/ew/d0ew00830c (visited on 01/31/2022).

[8] T. W. Kim, H. S. Yun, K. B. Kim, and S. B. Hong, "Development of real-time HABs detection technique using unmanned aerial vehicle (UAV)," *Journal of Coastal Research*, vol. 91, pp. 391–395, SI Aug. 1, 2019, ISSN: 0749-0208. DOI: 10.2112/SI91-079.1. [Online]. Available: https://doi.org/10.2112/SI91-079.1 (visited on 11/07/2021).

[9] M. Schwarzbach, M. Laiacker, M. Mulero-Pázmány, and K. Kondak, "Remote water sampling using flying robots," in *International Conference on Unmanned Aircraft Systems (ICUAS)*, May 2014, pp. 72–76. DOI: 10.1109/ICUAS.2014.6842240.

[10] C. Koparan, A. B. Koc, C. V. Privette, and C. B. Sawyer, "Autonomous in situ measurements of noncontaminant water quality indicators and sample collection with a UAV," *Water*, vol. 11, no. 3, p. 604, Mar. 2019. DOI: 10.3390/w11030604. [Online]. Available: https://www.mdpi.com/2073-4441/11/3/604 (visited on 11/07/2021).

[11] M. Chung, C. Detweiler, M. Hamilton, J. Higgins, J.-P. Ore, and S. Thompson, "Obtaining the thermal structure of lakes from the air," *Water*, vol. 7, no. 11, pp. 6467–6482, Nov. 2015. DOI: 10.3390/w7116467. [Online]. Available: https://www.mdpi.com/2073-4441/7/11/6467 (visited on 11/07/2021).

[12] M. Schiele and T. Letessier, "Amphibious drone field report, belize," Bertarelli Foundation, Marine Management Organisation and Turneffe Atoll Sustainability Association, May 1, 2019.

[13] "Hamadori 3000," - . (Feb. 15, 2022), [Online]. Available: http://www.selab.jp/products/hamadori3000/ (visited on 05/02/2023).

[14] J. Moore, A. Fein, and W. Setzler, "Design and analysis of a fixed-wing unmanned aerial-aquatic vehicle," in *IEEE International Conference on Robotics and Automation (ICRA)*, May 2018, pp. 1236–1243. DOI: 10.1109/ICRA.2018.8461240.

[15] W. Stewart, W. Weisler, M. MacLeod, *et al.*, "Design and demonstration of a seabird-inspired fixed-wing hybrid UAV-UUV system," vol. 13, no. 5, p. 056 013, Aug. 2018, ISSN: 1748-3190. DOI: 10.1088/1748-3190/aad48b. [Online]. Available: https://doi.org/10.1088/1748-3190/aad48b (visited on 11/16/2021).

[16] W. Weisler, W. Stewart, M. B. Anderson, K. J. Peters, A. Gopalarathnam, and M. Bryant, "Testing and characterization of a fixed wing cross-domain unmanned vehicle operating in aerial and underwater environments," *IEEE Journal of Oceanic Engineering*, vol. 43, no. 4, pp. 969–982, Oct. 2018, ISSN: 1558-1691. DOI: 10.1109/JOE.2017.2742798.

[17] H. Alzu'bi, I. Mansour, and O. Rawashdeh, "Loon copter: Implementation of a hybrid unmanned aquatic–aerial quadcopter with active buoyancy control," *Journal of Field Robotics*, vol. 35, no. 5, pp. 764–778, 2018, ISSN: 1556-4967. DOI: 10.1002/rob.21777. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/rob.21777 (visited on 04/11/2023).

[18] P. Rodrigues, F. Marques, E. Pinto, *et al.*, "An open-source watertight unmanned aerial vehicle for water quality monitoring," in *OCEANS 2015 - MTS/IEEE Washington*, Oct. 2015, pp. 1–6. DOI: 10.23919/OCEANS.2015.7404447.

[19] M. Hsu, "AquaFly: A tilt-rotor vertical take-off and landing aquatic unmanned aerial vehicle," M.A.S. University of Toronto (Canada), Canada, 2020, 104 pp. [Online]. Available: https://www.proquest.com/docview/2468374781/abstract/D0181CC0C8E4380PQ/1 (visited on 11/16/2021).

[20] D. Manoharan, C. Gajendran, M. Padmanabhan, S. Vignesh, S. Rajesh, and G. Bhuvaneshwaran, "Design and development of autonomous amphibious unmanned aerial vehicle for in situ water quality assessment and water sampling," *Journal of Unmanned Vehicle Systems*, vol. 9, no. 3, pp. 182–204, Sep. 1, 2021, Publisher: NRC Research Press, ISSN: 2291-3467. DOI: 10.1139/juvs-2020-

0036. [Online]. Available: https://cdnsciencepub.com/doi/abs/10.1139/juvs-2020-0036 (visited on 11/16/2021).

[21] B. Esakki, S. Ganesan, S. Mathiyazhagan, *et al.*, "Design of amphibious vehicle for unmanned mission in water quality monitoring using internet of things," *Sensors*, vol. 18, no. 10, p. 3318, Oct. 2018. DOI: 10.3390/s18103318. [Online]. Available: https://www.mdpi.com/1424-8220/18/10/3318 (visited on 11/16/2021).

[22] R. D. Eubank, "Autonomous flight, fault, and energy management of the flying fish solar-powered seaplane.," Thesis, 2012. [Online]. Available: http://deepblue.lib.umich.edu/handle/2027.42/91453 (visited on 04/12/2023).

[23] R. Eubank, E. Atkins, and S. Ogura, "Fault detection and fail-safe operation with a multiple-redundancy air-data system," in *AIAA Guidance, Navigation, and Control Conference*, ser. Guidance, Navigation, and Control and Co-located Conferences, American Institute of Aeronautics and Astronautics, Aug. 2, 2010. DOI: 10.2514/6.2010-7855. [Online]. Available: https://arc.aiaa.org/doi/10.2514/6.2010-7855 (visited on 04/12/2023).

[24] R. Zufferey, R. Siddall, S. F. Armanini, and M. Kovac, "Sailing and flying with a multimodal robot," in *Between Sea and Sky: Aerial Aquatic Locomotion in Miniature Robots*, ser. Biosystems & Biorobotics, R. Zufferey, R. Siddall, S. F. Armanini, and M. Kovac, Eds., Cham: Springer International Publishing, 2022, pp. 167–195, ISBN: 978-3-030-89575-4. DOI: 10.1007/978-3-030-89575-4_11. [Online]. Available: https://doi.org/10.1007/978-3-030-89575-4_11 (visited on 04/11/2023).

[25] G. Pisanich and S. Morris, "Fielding an amphibious UAV: Development, results, and lessons learned," in *Proceedings. The 21st Digital Avionics Systems Conference*, vol. 2, Oct. 2002, pp. 8C4–8C4. DOI: 10.1109/DASC.2002.1052944.

[26] B. Lou, G. Wang, Z. Huang, and S. Ye, "Preliminary design and performance analysis of a solar-powered unmanned seaplane," *Journal of Aerospace Engineering*, vol. 233, no. 15, pp. 5606–5617, Dec. 1, 2019, ISSN: 0954-4100. DOI: 10.1177/0954410019852572. [Online]. Available: https://doi.org/10.1177/0954410019852572 (visited on 11/16/2021).

[27] G. Aiello, K. P. Valavanis, and A. Rizzo, "Fixed-wing UAV energy efficient 3d path planning in cluttered environments," *Journal of Intelligent & Robotic Systems*, vol. 105, no. 3, p. 60, Jul. 1, 2022, ISSN: 1573-0409. DOI: 10.1007/s10846-022-01608-1. [Online]. Available: https://doi.org/10.1007/s10846-022-01608-1 (visited on 10/12/2022).

[28] R. D. Eubank, J. M. Bradley, and E. M. Atkins, "Energy-aware multiflight planning for an unattended seaplane: Flying fish," *Journal of Aerospace Information Systems*, vol. 14, no. 2, pp. 73–91, 2017. DOI: 10.2514/1.I010484. [Online]. Available: https://doi.org/10.2514/1.I010484 (visited on 11/16/2021).

[29] P. Váňa, J. Faigl, J. Sláma, and R. Pěnička, "Data collection planning with dubins airplane model and limited travel budget," in *European Conference on Mobile Robots (ECMR)*, Sep. 2017. DOI: 10.1109/ECMR.2017.8098715.

[30] K. Yang and S. Sukkarieh, "3d smooth path planning for a UAV in cluttered natural environments," in *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Sep. 2008, pp. 794–800. DOI: 10.1109/IROS.2008.4650637.

[31] H. V. Abeywickrama, B. A. Jayawickrama, Y. He, and E. Dutkiewicz, "Comprehensive energy consumption model for unmanned aerial vehicles, based on empirical studies of battery performance," *IEEE Access*, vol. 6, pp. 58 383–58 394, 2018, ISSN: 2169-3536. DOI: 10.1109/ACCESS.2018.2875040.

[32] A. Seewald, H. G. De Marina, H. S. Midtiby, and U. P. Schultz, "Mechanical and computational energy estimation of a fixed-wing drone," in *IEEE International Conference on Robotic Computing (IRC)*, Nov. 2020, pp. 135–142. DOI: 10.1109/IRC.2020.00028.

[33] O. D. Dantsker, M. Theile, and M. Caccamo, "A high-fidelity, low-order propulsion power model for fixed-wing electric unmanned aircraft," in *AIAA/IEEE Electric Aircraft Technologies Symposium (EATS)*, Jul. 2018, pp. 1–16.

[34] *ArduPilot*, version 4.3.5, Mar. 26, 2023. [Online]. Available: https://ardupilot.org/.

[35] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *The International Journal of Robotics Research*, vol. 30, no. 7, pp. 846–894, Jun. 1, 2011, ISSN: 0278-3649. DOI: 10.1177/0278364911406761.

[Online]. Available: https://doi.org/10.1177/0278364911406761 (visited on 01/08/2023).

[36] H. Butler, M. Daly, A. Doyle, S. Gillies, T. Schaub, and S. Hagen, "The GeoJSON format," Internet Engineering Task Force, Request for Comments RFC 7946, Aug. 2016. [Online]. Available: https://datatracker.ietf.org/doc/rfc7946 (visited on 04/17/2023).

[37] D. Hsu, T. Jiang, J. Reif, and Z. Sun, "The bridge test for sampling narrow passages with probabilistic roadmap planners," in *IEEE International Conference on Robotics and Automation*, vol. 3, Sep. 2003, 4420–4426 vol.3. DOI: 10.1109/ROBOT.2003.1242285.

[38] R. Eubank, E. Atkins, and D. Macy, "Autonomous guidance and control of the flying fish ocean surveillance platform," in *AIAA Infotech@Aerospace Conference*, American Institute of Aeronautics and Astronautics, Apr. 6, 2009. DOI: 10.2514/6.2009-2021. [Online]. Available: https://arc.aiaa.org/doi/10.2514/6.2009-2021 (visited on 11/17/2021).

[39] J. R. Dormand and P. J. Prince, "A family of embedded runge-kutta formulae," *Journal of Computational and Applied Mathematics*, vol. 6, no. 1, pp. 19–26, Mar. 1, 1980, ISSN: 0377-0427. DOI: 10.1016/0771-050X(80)90013-3. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0771050X80900133 (visited on 08/15/2023).

[40] L. Perron and V. Furnon, *OR-tools*, version 9.4, Aug. 11, 2022. [Online]. Available: https://developers.google.com/optimization/.

[41] T. R. Oke, *Boundary Layer Climates*, 2nd edition. Methuen & Co. Ltd, 1987, 460 pp., ISBN: 0-415-04319-0.

[42] J. Kuffner and S. LaValle, "RRT-connect: An efficient approach to single-query path planning," in *IEEE International Conference on Robotics and Automation. Symposia Proceedings*, vol. 2, Apr. 2000, 995–1001 vol.2. DOI: 10.1109/ROBOT.2000.844730.