Wind-Field Estimation and Curvature Continuous Path Planning for Low Altitude Urban Aerial Mobility

Jay Patrikar

CMU-RI-TR-20-30

August 11, 2020

The Robotics Institute
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA

Thesis Committee:
Dr. Sebastian Scherer, *chair*
Dr. Maxim Likhachev
Dr. David Wettergreen
Fahad Islam

Submitted in partial fulfillment of the requirements for the degree of Master of Science in Robotics.

To my Parents...

Abstract

Unmanned Aerial Vehicles (UAVs) operating in dense urban areas need the ability to generate wind-aware collision-free, smooth, dynamically feasible trajectories between two locations. The complex and high-rise structure of the modern urban landscape affects the wind flow around it which has a pronounced effect on the UAVs operating within the urban canopy. In this work, we propose a wind-field estimation method that uses in-situ onboard wind measurements from a UAV operating within the flow-field to solve the problem of predicting the inlet conditions for a Computational Fluid Dynamics simulation. Gaussian Process Regression is used as a surrogate forward propagation function to achieve onboard computation. Real flight test results using an onboard anemometer verify the efficacy of the wind-field estimation algorithm. The flight test shows that the 95% confidence interval for the difference between the mean estimated inlet conditions and mean ground truth measurements closely bound zero, with the difference in mean angles being between -3.680° and 1.250° and the difference in mean magnitudes being between -0.206 m/s and 0.020 m/s.

Given the knowledge of a spatially-varying but temporally-static wind field, we formulate a two point boundary value problem (BVP) that uses piece-wise continuous polynomial curvature spirals to connect states in a sampling based planner. Rather than solving the BVP nonlinear constrained optimization problem online, we propose the use of a trajectory library of precomputed solutions that may act as surrogate solutions for the underlying planner. We provide an empirical analysis to verify the feasibility of our approach. Comparative simulation results with a trochoid-based BVP planner in a realistic urban setting are used to showcase the competitive performance of the proposed path planning method. For the given simulation scenario, we could demonstrate a 93% success rate for the algorithm in finding a valid trajectory.

Acknowledgments

First and foremost, I would like to thank my advisor Dr Sebastian Scherer. Thank you for believing in me and for giving me the opportunity to be a part of the Airlab. Your infectious energy and can-do spirit have time and again motivated me to do my best. As I continue my academic journey with you, I look forward to exciting times ahead.

I would also like to express my deepest gratitude to Dr Constantine Samaras. As the DOE project PI, he has been a constant source of motivation and guidance throughout my studies. I would also like to thank the complete team on the project: Thiago Rodrigues, Arnav Choudhary, Jacob Feldgoise, Vaibhav Arcot, Bastian Wagner, Aradhana Gehlot, Sophia Lau. The project would not have achieved the results it has without this team.

I would like to take this opportunity to thank the supremely talented members of the AirLab for all the insights and discussions. My collaborators from the lab: Brady Moon and Vishal Dugar; working with each of you was an inspiring and rewarding experience.

Lastly, I would like to thank my family; Mom and Dad, this would not have been possible without you. Thank you for encouraging me to follow my dreams. To my sister and my two cute nephews; thank you for being a constant source of fun and energy.

Contents

When this dissertation is viewed as a PDF, the page header is a link to this Table of Contents.

List of Figures

Chapter 1

Introduction

This chapter presents the motivation for this thesis, gives the outline of the following chapters, and provides an acknowledgement of the contributions.

1.1 Motivation

The recent COVID-19 pandemic saw an increase in the use of robotic technologies. From crowd monitoring to medical and groceries deliveries, ground and aerial robots were seen plying city streets in an effort to control the pandemic and mitigate its effects on daily life [2]. While this exercise did provide a glimpse into a future where autonomous agents have an increased presence in our life, it also exposed many deep-seated flaws in the design of these agents. As robots left the factory and research floors, and were placed on the streets, many of the assumptions that work well within these controlled environments did not translate well in the real world [3]. Looking ahead, as the robots make this shift into the real world, the technological focus has steadily shifted from proof-of-concept prototypes to a public policy domain. Issues like safety of robots and humans need special consideration to enable widespread adoption of autonomous agents. One key idea is to adapt the algorithms and policies in a domain specific manner to ensure that as robots navigate specific arenas they are aware of the possible dangers and have tools and methods to handle these situations.

One arena under focus, which is also the focus of this study, are the dense urban areas which account for 55 % of the world's population [4]. Specifically, the study

focuses on the use of Unmanned Aerial Vehicles (UAVs) flying at low altitudes in dense urban canopies. Modern cityscapes provide a unique set of challenges for unmanned aerial vehicles. The presence of tall buildings on both sides of the streets creates a canyon-like situation inside urban areas. Generating feasible trajectories within these urban canyons is challenging because a) urban canyons are often densely packed with narrow passageways between adjacent buildings b) urban canyons affect the flow of air, and this leads to complicated and often dangerous wind patterns that may adversely affect UAV operations [5]. Both problems need to be explicitly addressed to generate feasible trajectories that ensure reliable and safe aerial operations. We thus wish to address the problem of wind-aware path planning for UAVs flying low altitude delivery or surveillance missions inside the urban canopy.

1.2 Outline

The thesis is organised as follows:

Chapter 2 defines the problem and provides details on various assumptions. Chapter 3 has a literature survey on the state-of-the-art methods. Section 3.1 covers the literature for estimating wind-fields and Section 3.2 covers wind-aware path planning. The Chapter also identifies research gaps and how the thesis contributes to the current research.

Chapter 4 focuses on the proposed wind-field estimation methodology. Section 4.2 provides details on the proposed methodology along with the mathematical formulations. Section 4.3 provides details on the field trials and the results. Section 4.4 presents the discussions and insights.

Chapter 5 focuses on developing a wind-aware path planning method. Section 5.2 formulates the broader path planning problem and introduces the symbols and kinematic model. Section 5.3 details the wind-aware BVP formulation. Section 5.4 focuses on the global sampling-based planner and Section 5.5 provides implementation details and a feasibility study. Section 5.6 provides simulation results and comparative results with a baseline. Section 5.7 presents the discussions and insights.

Chapter 6 identifies shortcomings in the current approaches and details some future improvements. Finally Chapter 7 presents overall conclusions.

1.3 Declaration of Contributions

The work in Chapter 4 was carried out in collaboration with Brady Moon. The work in Chapter 5 was carried out in collaboration with Vishal Dugar and Vaibhav Arcot. The work in this thesis has large overlaps with two accepted original contributions by these authors in conferences. The author of the thesis is also the first author on both the manuscripts.

1. Patrikar, Jay, Vishal Dugar, Vaibhav Arcot, and Sebastian Scherer. "Real-time Motion Planning of Curvature Continuous Trajectories for Urban UAV Operations in Wind" 2020 International Conference on Unmanned Aircraft Systems (ICUAS). IEEE, 2020.
2. Patrikar, Jay, Brady Moon, and Sebastian Scherer. "Wind and the City: Utilizing UAV-Based In-Situ Measurements for Estimating Urban Wind Fields." 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE, 2020.

Chapter 2

Problem Characterization

This chapter provides a high-level view of the specific problem we wish to address in this work and also identifies the key assumptions of our approach.

2.1 Problem Statement

Based on the motivations outlined in Section 1.1, we define our problem statement as:

Given a non-holonomic UAV with a start and goal location, plan a optimal collision-free, smooth, dynamically feasible path in a dense, cluttered, and windy urban environment in real-time.

The following subsections explore the reasoning behind the different parts of this problem statement.

2.1.1 Why consider a UAV to be non-holonomic?

It is trivial to see why a fixed-wing UAV or a Vertical Take-off and Landing (VTOL) UAV in cruise configuration can be treated as non-holonomic system but multi-rotors or VTOLs in hover configuration can exhibit both holonomic and non-holonomic behaviour depending on the path following strategy. Multirotors consume the most energy when they are hovering and as such a stop-to-turn strategy can easily become an inefficient method of path following. Hence, in the problem statement we choose to treat all the UAVs as a non-holonomic system.

2.1.2 What constitutes a smooth dynamically feasible path?

In their seminal work Lavalle et al. [6] defined the *kinodynamic planning* problem as ".. design a feasible open-loop trajectory that satisfies both global obstacle constraints and local differential constraints.". Over the years, the field has seen significant growth as detailed in Section 3.2 but achieving real-time performance from a kinodynamic planner that allows aggressive motions has been elusive.

Over the years, what constitutes "smoothness" has been revised. How smooth should a path be so that a non-holonomic system with a nominal tracking algorithm can follow it? This depends on the order of the differential constraints that the planner can respect. Dubins [7] identified that for a car that can only move forward with a given turn radius constraint, the shortest path between any two states belongs to one of six kinds of curves consisting of straight lines and constant curvature arcs. While such a simplified approach can generate paths that look feasible, these paths are in fact discontinuous in curvature space. This leads to large cross track errors as soon as the tracking speed increases beyond a certain threshold because realistically systems need a finite time to transition from one curvature state to the other. Recent works have pushed the definition of smoothness by constructing paths in "snap" space [8]. Such algorithms leverage the differentially flat dynamics of a multi-rotors to generate analytical expressions for control inputs that are used to propagate forward dynamics. But these algorithms rely on predefined waypoints which does not make them purely real-time and collision-free planners.

As a middle ground, we propose in our problem statement generating a smooth path that respects constraints on curvature and curvature-rate for a generic unicycle non-holonomic system. By constructing curvature continuous paths we can decrease the cross-track errors while following the path at higher speeds. We argue that these constraints lead to a path that is smooth enough for the kind of operations that are in the purview of this work.

2.1.3 Why should we care about wind in path planning?

As previously mentioned, wind in urban canopies can have adverse effects on the UAV operations. More specifically, wind affects the practically achievable curvature rate of a UAV. Consider a case of a fixed-wing UAV that is following a Dubins-type optimal path with maximum curvature turns. This maximum curvature can only be achieved if the UAV holds its maximum safe roll attitude. Now, in a situation that the UAV is turning into headwind, if it holds the same maximum safe roll attitude, the achievable maximum curvature in inertial space is lower than what is needed to follow the wind-agnostic curvature optimised path. This will lead to a larger than expected tracking error. On the other hand, in a situation that the UAV is turning into tailwind, the maximum curvature can be archived without hitting the maximum safe roll attitude, thus under-utilizing the system capability to achieve higher than expected curvature in inertial space. With obstacles so close to the flight paths, a large unexpected tracking error can result in unsafe paths. Thus, in our problem statement we explicitly incorporate the effect of wind on the turn radius of the system thereby reducing the burden on the path following algorithm to maintain a low tracking error in the presence of wind while maximising utility.

For the scope of this work we consider the wind to be known precisely as spatially varying but temporally static map. Future work will look into addressing these assumptions.

2.2 Solution Strategy

We choose an solution strategy in which we identify two challenges that need to be tackled to address the problem statement.

Challenge 1. Urban Wind-Field Estimation: In order to solve a wind-aware path planning query, the UAV needs to have the knowledge of the underlying wind conditions in the city. We explore the use of in-situ wind measurements from a UAV to predict this spatially-varying but temporally-static wind field.

Challenge 2. Wind-Aware Curvature Continuous Path Planning Assuming a known wind-field, we use it to solve a path planning query in realtime that produces smooth, curvature continuous, collision-free, dynamically feasible paths.

We formulate a two-point boundary value problem (BVP) to be used in nonlinear optimization routine that can connect states in a sampling-based planner.

2.3 Key Assumptions:

Following are the prominent assumptions of the approach:

1. We assume that a fairly accurate high resolution static map of the city environment is available. The assumption holds as the urban landscape does not change drastically and that LiDAR maps are available for most cities.
2. Fairly accurate wind measurements are available from onboard sensors. We use an onboard ultrasonic anemometer to satisfy this assumption but any algorithm that provides a good estimate can be used.
3. Wind-field is spatially-varying but temporally static. Such a quasi-static approximation holds when the wind conditions do not change rapidly.

Chapter 3

Background

Wind-field estimation and wind-aware path planning have been previously studied in literature. This chapter provides a literature review on the current methods and also explores the gaps and drawbacks in state-of-the-art methods.

3.1 Wind-field Estimation

UAVs are uniquely positioned to act as a source of in-situ wind measurements. It has been shown that UAVs, using only their onboard attitude and position sensors, may serve as a source of reliable local wind measurements [9, 10] . Our work explores the possibility of using these local wind estimates to predict a global wind field. Wind flow estimation to enable autonomous dynamic soaring [11] has predominantly focused on estimating high altitude, clutter-free wind fields. Methods using Gaussian Process Regression [11], polynomial parameterization of the wind field [12], and Weibull probability density function with Prandtl's power law relationship [13] have been presented. Low altitude urban wind fields by comparison are more complicated and difficult to predict.

Computational Fluid Dynamics (CFD) has been presented as a promising solution to calculate the low altitude urban wind flow patterns. Techniques using Reynolds-Averaged Navier–Stokes (RANS) solvers have gained prominence over others, mainly due to their computational advantage. Using CFD results for UAV path planning has been explored in some works [14, 15], but they lack a comprehensive analysis

on how close the simulation models match the real world conditions. One notable effort is made by Ware et al. [16] who validated the CFD simulation results using in-situ measurements from ground stations. The work however uses historical data for estimating the boundary conditions which may not lead to sufficiently accurate wind field predictions.

Improving the predictive accuracies of CFD simulations is an active field of research [17]. Previous work on Uncertainty Quantification (UQ) [18] has proven that uncertainties in inlet conditions have a higher effect on the accuracy of CFD results than other parameters such as aerodynamic roughness. Other works [19] that compared CFD results with real world data have argued that even for minor changes in the inlet wind direction the simulated flow patterns can change considerably, to the extent that planning decisions might be influenced. It is thus important to have a reliable estimate of the inlet conditions. One obvious solution is to use wind sensors at the periphery of the urban area or a tall structure within the city. However, measurements from these sources can introduce considerable error as often times it is difficult to determine the errors between the incoming wind at the boundary and measured wind conditions. Jorge Sousa et al. [20] have addressed this problem of predicting the distribution of the boundary conditions (inlet speed and angle) by solving a inverse problem of estimating inlet conditions using an ensemble Kalman filter (EnKF) and given in-situ measurements from static sensors. Their subsequent work provides real world validation [21] of the methods.

Our approach builds on the previous work by Jorge Sousa et al. [20] and extends it to incorporate dynamic sensors. While their approach relies on an EnKF to solve the inverse problem, we propose using a Particle Filter. While both approaches are shown to achieve similar performance given a large enough ensemble size [22], EnKFs are often more suitable for problems with high dimensionality. But EnKF has a tendency to provide approximations to the posterior that are too close to a Gaussian distribution and as such Particle Filters which directly sample from the exact posterior distribution can lead to better accuracies [23]. One other point of difference is that while Sousa's approach used a polynomial chaos expansions as a surrogate function, we use Gaussian Process Regression models as they were found to be more appropriate for the spatially varying distributions.

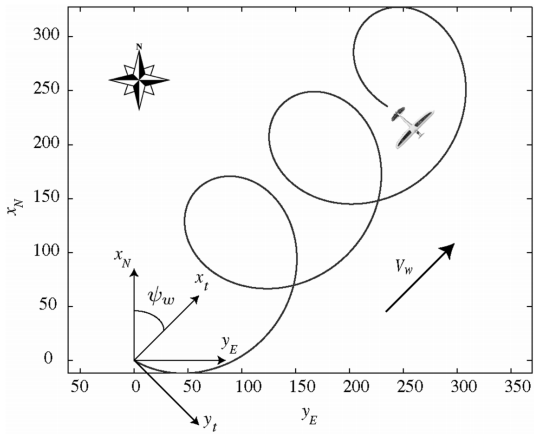


Figure 3.1: Trochoidal path and trochoidal frame. [1]

3.2 Wind-Aware Path Planning

Wind-aware path planning has a rich history with mutiple school of thoughts. One possible way to address the issue is to perform wind-agnostic path planning but design a controller that is capable of handling wind disturbances [24][25]. While these may be intuitive solutions, the systems may produce a large cross-track error if the wind on the planned route leads to actuator saturation. Recent works [26] have tried to address these issues by incorporating actuator saturation in the path following algorithms. One recent approach [27] uses a strategy that switches to a provably-safe controller if the vehicle violates a pre-calculated tracking error bound. This bound captures all possible deviations due to external disturbances like the wind but may lead to a conservative solution. Another approach [28] uses an estimate of the wind and the corresponding drag forces to generate trajectories given a set of waypoints. All of these approaches assume a path planning algorithm for waypoint generation. Our work explores the possibility of explicitly including wind in the path planning phase, thereby reducing the burden on the lower level controller. The current approach derives its focus from two broad areas: path planning in vector fields and kinodynamic path planning.

Path planning in a vector field like ocean currents or wind is an established problem in robotics. Subramani et al. [29] used level-set PDEs to find a time-optimal path in ocean current but ignore explicit vehicle dynamics as underwater operations

are often at lower speeds. There exists a large body of work [30, 31] for calculating energy-optimal paths for autonomous dynamic soaring, but the considered arena is often high altitude obstacle-free airspace. There is a recent interest in solving for drone trajectories in low altitude cluttered urban environments [16, 32, 33, 34]. However, none of the methods take into account wind and dynamic feasibility.

In the context of kinodynamic path planning for mobile robots, recent popular approaches use a probabilistically complete sampling-based planner like BIT* [35] or RRT* [36] that provide asymptotic optimality guarantees. All of these planners need to solve a two-point boundary value problem (BVP) to connect pairs of states with dynamically feasible trajectories. Solving BVPs is often too slow for real-time performance. Methods like neighborhood classification [37], Sequential Quadratic Programming (SQP) formulation [38] , bang-bang control based trajectory generation procedure [39] or resolution-complete online-offline trajectory library pre-computation [40] are used to achieve a speed-up in the planning process but none of them take into account wind. One approach is to avoid solving the BVP by using a forward propagation model [41] but execution times are arguably too slow for real-time.

Trochoids are the natural wind-aware extension of Dubins-type paths and have been explored in literature as a BVP solution to connect two states in the presence of wind [1, 42, 43, 44]. While an intuitive solution, trochoids are discontinuous in curvature space. Following a curvature discontinous path at high speeds may lead to excessive path tracking errors owing to the fact that dynamical systems like UAVs require a finite time interval to achieve constant curvature trim states. With this limitation in mind but given their wide usage, in the current work, we use trochoids as a baseline BVP solution for comparison.

In our work, we build on the previous approaches that aim to solve a BVP to connect states in a sampling based planner. We formulate the BVP as a wind-aware non-linear constrained optimization problem by using polynomial spirals [45] with constraints not only on curvature but also curvature rate.

Chapter 4

Wind-Field Estimation

A high-quality estimate of wind fields can potentially improve the safety and performance of Unmanned Aerial Vehicles (UAVs) operating in dense urban areas. Computational Fluid Dynamics (CFD) simulations can help provide a wind field estimate, but their accuracy depends on the knowledge of the distribution of the inlet boundary conditions. This Chapter presents a real-time methodology using a Particle Filter (PF) that utilizes wind measurements from a UAV to solve the inverse problem of predicting the inlet conditions as the UAV traverses the flow field. A Gaussian Process Regression (GPR) approach is used as a surrogate function to maintain the real-time nature of the proposed methodology. Real-world experiments with a UAV at an urban test-site prove the efficacy of the proposed method.

4.1 Approach

Enabling autonomous unmanned aerial vehicles (UAVs) to estimate wind patterns is crucial if they are to realize their potential in goods mobility, monitoring, and surveillance tasks in dense urban landscapes. Owing to their small size, low speeds, and close proximity to obstacles, the safety and operational performance of UAVs flying within cities is significantly affected by the prevailing wind conditions [5, 15]. An estimate of the wind field can provide planning and decision algorithms with necessary information to compute safer [46] and energy efficient paths [16].

While CFD is used to solve the forward problem of estimating wind conditions

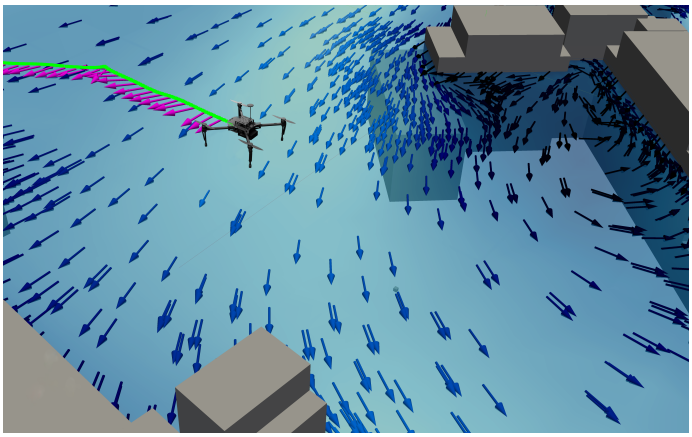


Figure 4.1: A CFD wind field using the estimated inlet boundary condition at the test location, along with the path of the UAV (green), and the corrected wind measurements (magenta) from the onboard anemometer.

given the knowledge of boundary conditions, this work explores the possibility of using UAV-based sequential local wind measurements using an onboard anemometer to solve the inverse problem of estimating the inlet boundary conditions using a particle filter (PF) based approach. A PF is known to provide a better estimate of the posterior belief if the relationship between the state and observed data is nonlinear, for which the application of the EnKF is not appropriate [47]. UAVs have an ability to fly in the roughness layer [14] that is just above the Urban Canopy Layer (UCL) and thus can potentially give a much better estimate of the inlet conditions affecting the UCL. Our setup also avoids the need of setting up a static wind measurement network, which may prove prohibitive because of cost and maintenance hurdles. As wind enters the city, it flows between structures and form a distinctive pattern. The core idea of our approach is to identify this pattern using the onboard measurements and then solve the inverse problem to estimate the inlet conditions. If the estimated boundary conditions are used as an input for running the CFD simulation, the resulting field should most closely resemble the actual wind field. In lieu of onboard measurements from the UAV using standard onboard sensors (GPS, IMU, ..), we use an ultrasonic anemometer. The contributions of this chapter are three fold:

1. We present a particle filter approach to solve the inverse problem of estimating the boundary conditions given sequential in-situ wind measurements from a UAV.

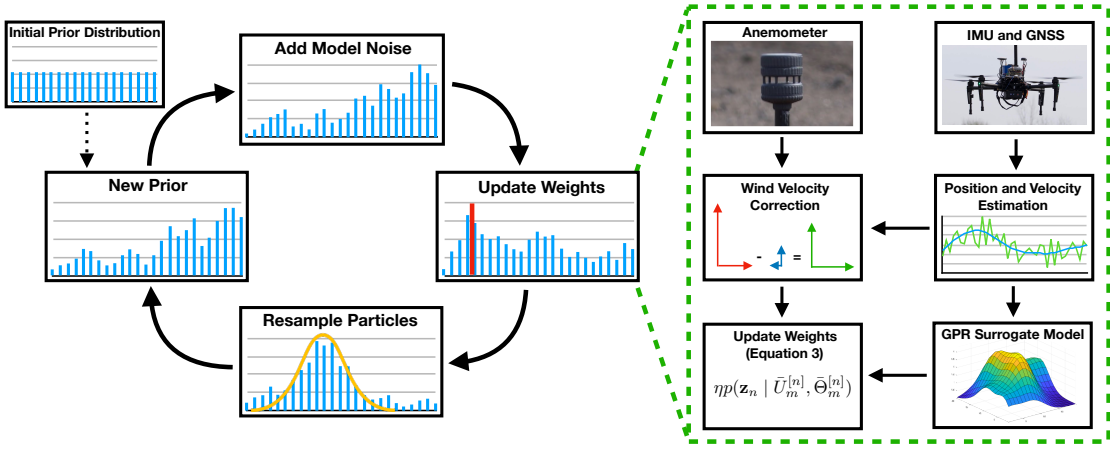


Figure 4.2: Overview: Wind measurements from an onboard anemometer are corrected for motion and bias. These measurements are then used in a Particle Filter framework to provide a posterior distribution on the inlet conditions by comparing with estimates from a Gaussian Process based forward propagation model.

2. We present a domain specific treatment to circumvent the problem of running multiple forward simulations using Gaussian process regression to maintain the real-time onboard nature of the methodology.
3. We present real world test results that prove the efficacy of the presented methodology.

The chapter is organized as follows: Section 4.2 provides details on the proposed methodology along with the mathematical formulations. Section 4.3 provides details on the field trials and the results. Section 4.4 presents the discussions and insights.

4.2 Methodology

This section details the methodology to estimate the boundary conditions given measurements from the UAV. Care was taken that the onboard algorithms are computationally tractable for any onboard computer with modest specifications. We assume the UAV to be operating at a constant altitude, but the presented method can be generalised to accommodate a variable altitude. The goal is to find the distribution on the wind inlet angle Θ and wind inlet magnitude U. For the case under consideration, the local wind angle θ and the local reduced velocity $u_r = \frac{u}{U}$ are

only a function of the inlet wind angle where u is the local wind magnitude. The methodology is summarized in Figure 4.2.

4.2.1 Mathematical Formulations

Mathematically, the problem can be stated as: Given a set of measurements $\mathbf{z_i} = \{\theta_i, u_i\}$ for $i \in [0, N]$ from the UAV such that θ_i represents the local wind angle, and u_i represents the local wind magnitude at the i^{th} data-point, find the belief $bel(\Theta_{0:i}, U_{0:i}) = p(\Theta_{0:i}, U_{0:i} \mid \mathbf{z_{1:i}})$ for $i \in [0, N]$ where Θ_i is the wind inlet boundary angle and U_i is the wind inlet magnitude after i measurements. To solve for this belief, we will perform a domain specific derivation of a particle filter approach [48].

Using the Bayes Rule we can write:

$$\begin{aligned} bel(\Theta_{0:i}, U_{0:i}) &= p(\Theta_{0:i}, U_{0:i} \mid \mathbf{z}_{1:i}) \\ &= \eta p(\mathbf{z}_i \mid \Theta_{0:i}, U_{0:i}, \mathbf{z}_{1:i-1}) p(\Theta_{0:i}, U_{0:i} | \mathbf{z}_{1:i-1}) \end{aligned} \tag{4.1}$$

Using the Markov property for conditional independence:

$$\begin{aligned} bel(\Theta_{0:i}, U_{0:i}) &= \eta p(\mathbf{z}_i \mid \Theta_i, U_i) p(\Theta_{0:i}, U_{0:i} | \mathbf{z}_{1:i-1}) \\ &= \eta p(\mathbf{z}_i \mid \Theta_i, U_i) p(\Theta_{0:i-1}, U_{0:i-1} | \mathbf{z}_{1:i-1}) \\ &\qquad p(\Theta_i, U_i | \Theta_{0:i-1}, U_{0:i-1}, \mathbf{z}_{1:i-1}) \\ &= \eta p(\mathbf{z}_i \mid \Theta_i, U_i) p(\Theta_{0:i-1}, U_{0:i-1} | \mathbf{z}_{1:i-1}) \\ &\qquad p(\Theta_i, U_i | \Theta_{i-1}, U_{i-1}) \end{aligned} \tag{4.2}$$

Thus, for the m^{th} particle, the weight can be calculated as

$$w_m = \eta p(\mathbf{z}_i \mid \Theta_i, U_i) \tag{4.3}$$

This probability distribution can be represented as a multivariate normal distribution:

$$p(\mathbf{z}_i \mid \Theta_i, U_i) = \frac{\exp -\frac{1}{2}(\mathbf{z}_i - \bar{\mathbf{z}}_i)^T \Sigma^{-1} (\mathbf{z}_i - \bar{\mathbf{z}}_i)}{\sqrt{(2\pi)^2 \mid \Sigma \mid}} \tag{4.4}$$

Where

$$\bar{\mathbf{z}}_\mathbf{i} = f(\Theta_i, U_i, \mathbf{X_i}) \tag{4.5}$$

where $\mathbf{X_i} = \{x_i, y_i\}$ are the coordinates of the UAV when the measurement $\mathbf{z_i}$ was taken. This function is the forward model and requires running the CFD solver, but given that it is a computationally expensive operation, we instead use a computationally cheaper but less accurate surrogate function. The details are further discussed in Section 4.2.3

4.2.2 Particle Filter

The particle filter implementation is detailed in Algorithm 1. M particles are initialized from a uniform distribution *UniSample* between minimum and maximum values. Each particle is a tuple of inlet angle and magnitude. The algorithm has three major components. The propagation step is the first step (Line 3), which in this case only involves adding zero mean Gaussian noise to the particles. This helps to avoid premature convergence. The next step (Line 5) involves calculating the weight of each particle based on how likely the particle represents the actual value of the inlet conditions. This weight is calculated for each particle using Equations 4.5 & 4.4. Finally based on the calculated weights, a re-sampling algorithm *ReSample* chooses particles for the next iteration based on the weighted probability distribution. The resampling process accounts for the difference of the target and the proposal distribution. To improve performance, we use a low-variance resampling strategy [48] to sample the distribution, as it helps preserve the diversity if samples have same importance factors and is a more systematic way for re-sampling than the independent random sampler. The sampler operates by choosing a random number between $[0, M^{-1}]$ and then repeatedly adds M^{-1} to the random number while selecting particles that correspond to the resulting summation. For the sake of brevity, the complete algorithm is not presented, and the reader is directed to Table 4.4 [48] for a detailed explanation.

4.2.3 Surrogate function

Calculating Equation 4.5 can prove to be a computationally expensive process, as it involves running a forward pass of the CFD simulation for the particular inlet condition and probing the location for the wind angle and magnitude. One way to avoid this issue, is to use a surrogate function to approximate these values. Jorge Sousa

Algorithm 1 Particle Filter

function PF($\mathbf{z_{1:N}}$, $[\{x_1, y_1\}, \ldots, \{x_N, y_N\}]$,M)
  $(U^{[0]}_{0:M}, \Theta^{[0]}_{0:M}) \leftarrow$ UniSample([0,U_{max}],[0,2π],M)
  for all n $\leftarrow$ 1:N **do**
    $(\bar{U}^{[n]}_{0:M}, \bar{\Theta}^{[n]}_{0:M}) = (U^{[n-1]}_{0:M}, \Theta^{[n-1]}_{0:M}) + \mathcal{N}(0, \sigma)$
    for all m$\leftarrow$ 1:M **do**
      $w_m = \eta p(\mathbf{z}_n \mid \bar{U}^{[n]}_m, \bar{\Theta}^{[n]}_m)$
      $(U^{[n]}_{0:m}, \Theta^{[n]}_{0:m}) \leftarrow$ ReSample($\bar{U}^{[n]}_{0:m}, \bar{\Theta}^{[n]}_{0:m}$, $w_{0:m}$)
    end for
  end for
  return $(U^N_{0:M}, \Theta^N_{0:M})$
end function

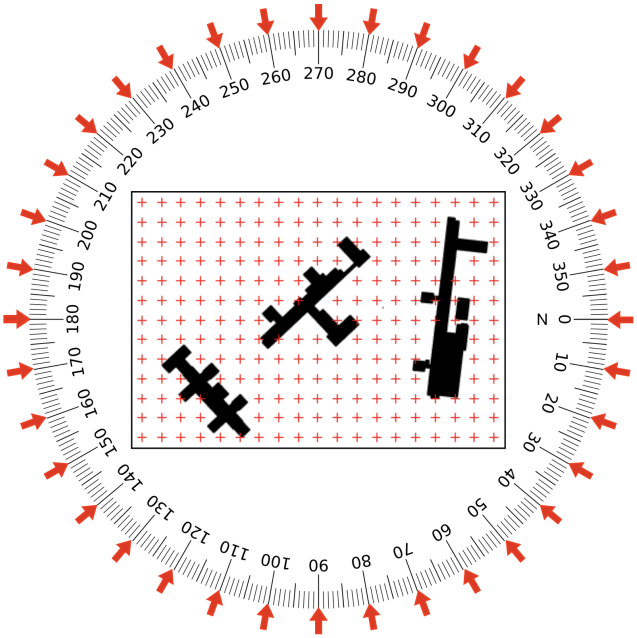


Figure 4.3: Data points (locations and inlet angles) for GPR training.

et al. [20] use a set of CFD simulations to construct polynomial chaos expansions (PCE) for the quantities of interest. For our use case, the PCE method gave a less accurate result, as the function variables also include the location along with the inlet angle. We decided to use a Gaussian Process Regression (GPR) to model the surrogate functions, as they have been shown to better represent spatial data. Two GPR models are trained: one for the local wind angles $\hat{f}_\theta$ and one for local wind reduced velocity $\hat{f}_{u_r}$. Both are functions of the inlet angle and probe location. Thus Equation 4.5 is replaced by:

$$\begin{aligned} \bar{u}_i &= \hat{f}_{u_r}(\Theta_i, \mathbf{X}_i) U_i \\ \bar{\theta}_i &= \hat{f}_\theta(\Theta_i, \mathbf{X}_i) \end{aligned} \tag{4.6}$$

The datapoints used to train the GPR models were obtained by running multiple CFD simulations with regularly spaced inlet angles and recording data on an equally spaced grid. The data-points are visually represented in Figure 4.3. The performance of the GPR models is discussed in Section 4.3.2.

4.2.4 Wind Measurements and Correction

Wind measurements are obtained from an ultrasonic anemometer onboard the UAV. The anemometer records the wind angle and magnitude with respect to the moving drone. Previous works that have used anemometers for onboard wind measurements [49, 50, 51] have reported that while the wind angle measurements were found to be accurate for all flight conditions, the magnitude measurements show a bias that reports a higher wind magnitude than expected. We found that the effect is more pronounced at lower UAV speeds. Thus, in order to record reliable wind measurements we need to remove the magnitude bias in addition to correcting for the UAV motion. In order to remove the magnitude bias we use the data collected by Brushi et al. [50] of a quadrotor flying with a ultrasonic wind sensor inside a wind tunnel. Given the similarity between the setup, the results in [50] are used to construct a polynomial mapping between measured and actual wind speeds. As the data lacks points in the lower range of values, it is augmented with results from our own setup. The UAV was flown in hover and constant ground speed mode for an extended amount of time in near-zero wind conditions. The final correction polynomial and ego-motion correction

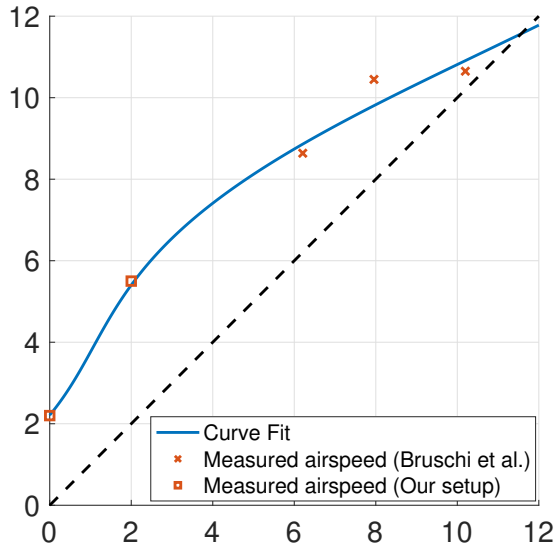


Figure 4.4: Figure shows the measured wind magnitude data-points and the fitted calibration curve.

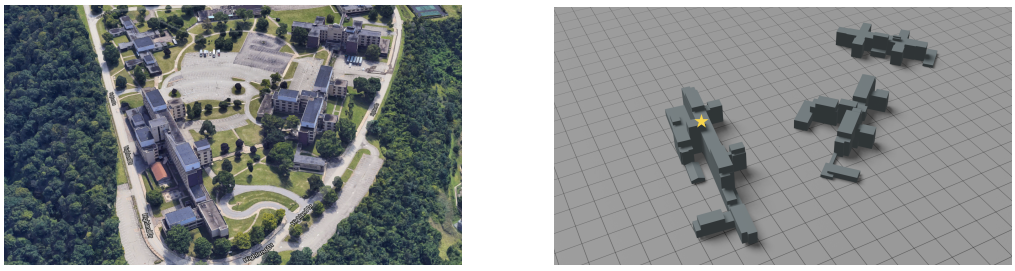
is given by:

$$
\begin{aligned}
w_m^{[actual]} &= -0.002(w_m^{[raw]})^4 + 0.052(w_m^{[raw]})^3 - 0.421(w_m^{[raw]})^2 + 1.917 w_m^{[raw]} - 2.7 \\
w_N &= (w_m^{[actual]} - \alpha)\cos(w_\theta^{[raw]}) - V_N \\
w_E &= (w_m^{[actual]} - \alpha)\sin(w_\theta^{[raw]}) - V_E \\
w_m^{[corrected]} &= \sqrt{w_N^2 + w_E^2} \\
w_\theta^{[corrected]} &= \arctan(w_E, w_N)
\end{aligned}
\tag{4.7}
$$

where V_N and V_E represent the inertial speed of the UAV in North and East directions, and w_N and w_E represent the corrected wind components in North and East directions.

4.2.5 Summary

The methodology assumes that an accurate 3D model of the environment is available, and the UAV has a sensor suite capable of reliably measuring the local wind angle and magnitude. The steps are as follows:



(a) (b)

Figure 4.5: (a) Image of the field validation test site. (b) 3D model of test site. Yellow star represents the location of the ground truth weather station on the top of the building.

1. Given the 3D model, run multiple forward RANS CFD simulations to collect data points (Fig. 4.3) for training the surrogate GPR models represented in Equation 4.6.
2. As the UAV is flying in the flow field, collect the raw wind measurements and correct them using Equation 4.7.
3. Use the offline trained surrogate models and the corrected wind measurements to estimate the inlet conditions using Algorithm 1.

4.3 Field Validation

This section gives details on the test setup and the hardware used for field validation of the algorithm. Discussion on the experiments and results follows. Field trials involve autonomously flying a predetermined sequence of waypoints and estimating the inlet conditions. For validation, the estimated inlet conditions are compared to a static wind sensor mounted on a rooftop nearby. The static wind sensor is measuring the free-stream wind magnitude and direction.

4.3.1 Setup

Test site

Our chosen test site is located on an elevated piece of land with around ten buildings, all ranging in heights of 5 to 35 meters (H_{max}). The buildings are abandoned and only authorised personnel are allowed within the area. The complex of buildings effectively replicates a typical urban environment and will allow for complex interactions of the wind and buildings. Because the test site is elevated from the surroundings, it provides an ideal environment to minimize disturbances to the inlet conditions.

An image and model of the test site are seen in Figure 4.5a and 4.5b respectively. The size of the site is roughly 300×250 meters. To serve as a ground truth for the inlet conditions, a weather station was placed in the location marked with a star in Figure 4.5b on top of the tallest building, .

Hardware setup

Our UAV test platform is a DJI M100 quadrotor with a FT Technologies FT205EV Lightweight Ultrasonic Wind Sensor. This wind sensor is ideal for mounting on a flying platform because of its low weight and integrated magnetometer. To minimize disturbances from the quadrotor propellers, the sensor is mounted on a 40 cm long carbon fiber pole. Figure 4.6 shows the UAV with the sensor. The DJI SDK fuses GPS and IMU data to provide the position and velocity estimates.

For our static weather station we used a Maximet GMX500 which has a resolution of 0.01 m/s and an accuracy of $\pm 3\%$. From the station we recorded wind heading and wind magnitude, as well as the UTC time, using a single board computer. The inlet estimates from the UAV and the static measurements are synced using the GPS time on the static sensor. To reduce the high frequency noise, all the sensor measurements are passed through a median filter before using them in the algorithms.

Figure 4.6: The test UAV with the ultrasonic wind sensor.

4.3.2 Preprocessing

GPR Models

In order to successfully test the methodology, GPR models need to be trained to estimate the wind angles and reduced velocities given the inlet angles for the area under consideration. To generate the training data, we run CFD simulations on regularly spaced inlet angles spanning from 0° to 360° with a step size of 10°. To sample points spatially, a regularly spaced 20 m resolution grid in X and Y was used as probe locations. Figure 4.3 shows these points in red. Care was taken to remove points that fall within structures. CFD simulations were carried out using the OpenFoam `SIMPLE` solver [52]. The mesh was created using SnappyHexMesh. The CFD assumes in-compressible flows, and the values are recorded at steady state. A standard $\kappa - \epsilon$ model is used to model the turbulence characteristics.

The GPR models use a radial basis function (RBF) with a length scale of 10 as the kernal function. The training is carried out using the `sklearn` library [53]. To test the model performance, we plot both the angle and reduced magnitude predictions against CFD results in Figures 4.9 and 4.11 respectively.

Correlation analysis

To identify which areas in the map are the most informative, the correlation between local wind angles and inlet angles was evaluated across the test space. For each point

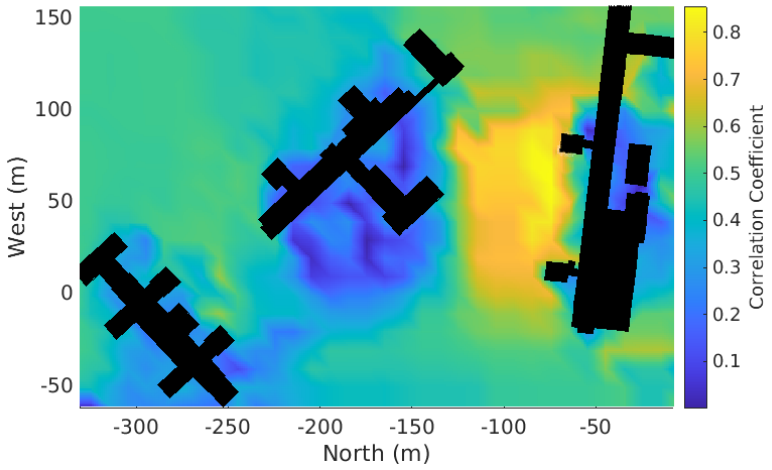


Figure 4.7: Correlation between local wind angles and inlet wind angles.

in a five-meter spaced grid, a Pearson correlation coefficient was calculated between the local wind angles and inlet angles. Figure 4.7 shows the correlation coefficients across the test space. This shows how areas which are often blocked from the wind will not correlate well to the inlet angle, and taking measurements there would not be as beneficial for inlet condition estimation. In contrast, there are also areas with high correlation, which are usually areas where the wind will be less hindered regardless of the inlet conditions. In the case of our test site, the center has the highest correlation, as well as the spaces farther out from the buildings.

4.3.3 Experiments

Experiments were carried out over a month, and test days were chosen based on the wind forecasts. A set of waypoints for the UAV path were selected based on the correlation graph. The resulting path can be seen in Figure 4.8. The commanded altitude of the path was 20 meters AGL, and it was flown at a commanded inertial speed of 2 m/s. The path was flown autonomously, and the corrected wind angle and speed are shown in Figures 4.9 and 4.11 respectively. The data from the the weather station was recorded concurrently for comparison. In order to match the training of the surrogate function, the wind measurements while the UAV was on the ground, taking off, and landing were not used in the particle filter, but rather only

the measures when the UAV was at the commanded flight path altitude.

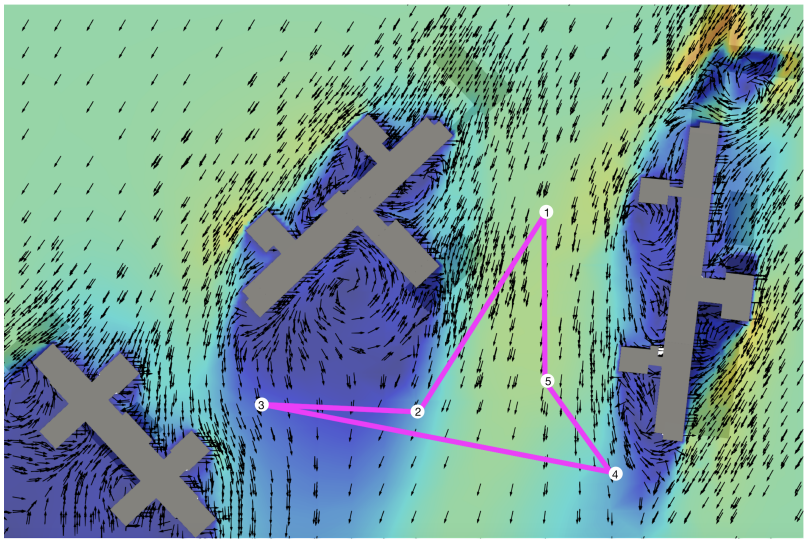


Figure 4.8: The path followed autonomously by the UAV at the test site. The waypoints are numbered in the order they were visited.

4.3.4 Results

The estimated inlet angle and magnitude from one of the test runs are plotted in Figures 4.10 and 4.12 respectively. The figures show that the estimated inlet conditions are in agreement with the values seen by the onboard anemometer. The figures also show the mean and one standard deviation of the ground truth measurements. As comparison, the cumulative means and one standard deviation of the PF output is plotted. It is clear that the mean of the PF outputs converge to the mean of the ground truth measurements. The 95% confidence interval for the difference between the mean estimated inlet angle and mean ground truth angle is $-3.680°$ and $1.250°$. The 95% confidence interval for the difference between the mean estimated inlet magnitude and mean ground truth magnitude is $-0.206\ m/s$ and $0.020\ m/s$. Zero lies within the bounds of both confidence intervals, thus showing the statistical significance of these results.

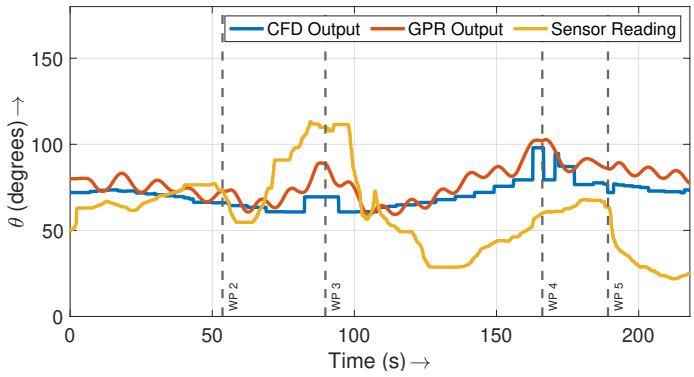


Figure 4.9: The motion and bias corrected local wind angle readings for the test run. Also shown is the CFD and GPR output for the same locations as the test run using the mean of the inlet ground truth measurements.

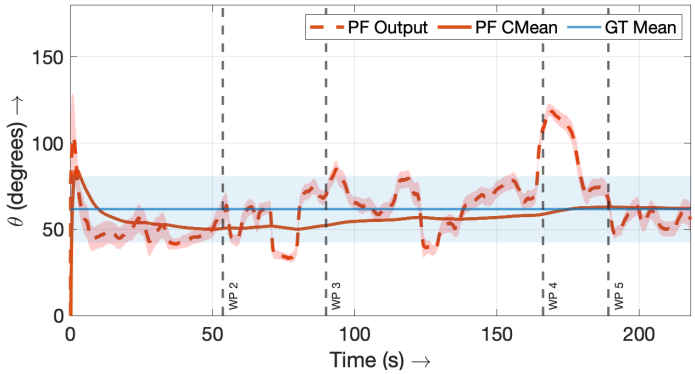


Figure 4.10: The wind inlet angle (and one std. deviation) estimated from the particle filter and the mean inlet angle (and one std. deviation) measured by the ground truth weather station.

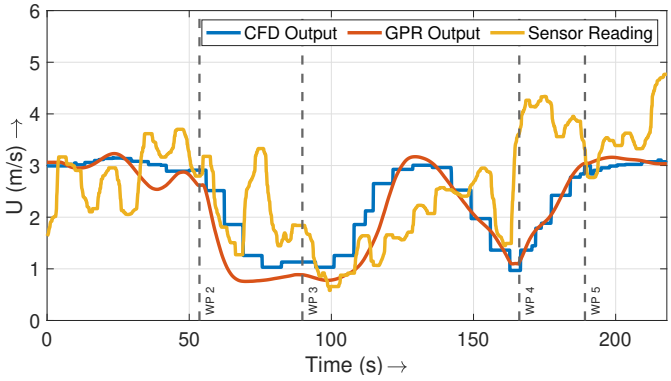


Figure 4.11: The motion and bias corrected local wind magnitude readings for the test run. Also shown is the CFD and GPR output for the same locations as the test run using the mean of the inlet ground truth measurements.

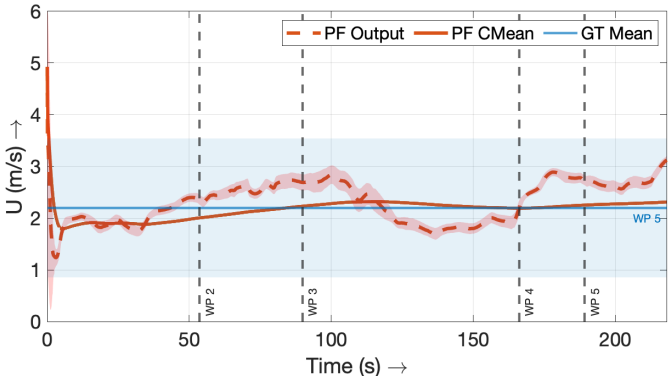


Figure 4.12: The wind inlet magnitude (and one std. deviation) estimated from the particle filter and the mean inlet magnitude (and one std. deviation) measured by the ground truth weather station.

4.4 Chapter Insights

A Particle Filter based method to use onboard wind measurements from a UAV to solve the inverse problem of estimating the inlet conditions to improve CFD-based wind field estimation is presented. Following are the major insights:

1. The work shows that it is possible to reliably use an anemometer on a moving UAV platform as sources of local wind measurements.
2. A Gaussian Process Regression model can be used as a surrogate function in-lieu of a CFD forward propagation to achieve real-time performance.
3. The results are shown to be consistent with the measurements from a static roof mounted weather station.

Chapter 5

Wind-Aware Curvature Continuous Path Planning

A key challenge in enabling autonomous Unmanned Aerial Vehicles (UAVs) to operate in cluttered urban environments is to plan collision-free, smooth, dynamically feasible trajectories between two locations with the wind in real-time. This chapter presents a novel path planning strategy using sampling-based planning that uses a two-point boundary value problem (BVP) to connect states in the presence of wind. Unlike most approaches that use a curvature discontinuous solution, the proposed BVP is formulated as a nonlinear constrained optimization problem with curvature and curvature-rate continuous profile to generate smoother trajectories. To achieve real-time performance, our method uses surrogate solutions from a pre-calculated library while solving the planning problem and then runs a repair routine to generate the final trajectory. To validate the feasibility of the offline-online strategy, simulation results on a 3D model of an actual city block with a realistic wind-field are presented. Results with a trochoid-based BVP solver are also presented for comparison.

5.1 Approach

In this chapter, we present a novel real-time method to solve wind-aware curvature-smooth two-point boundary value problems to connect states in a sampling-based global planner. The planner assumes a known map and a known spatially varying but

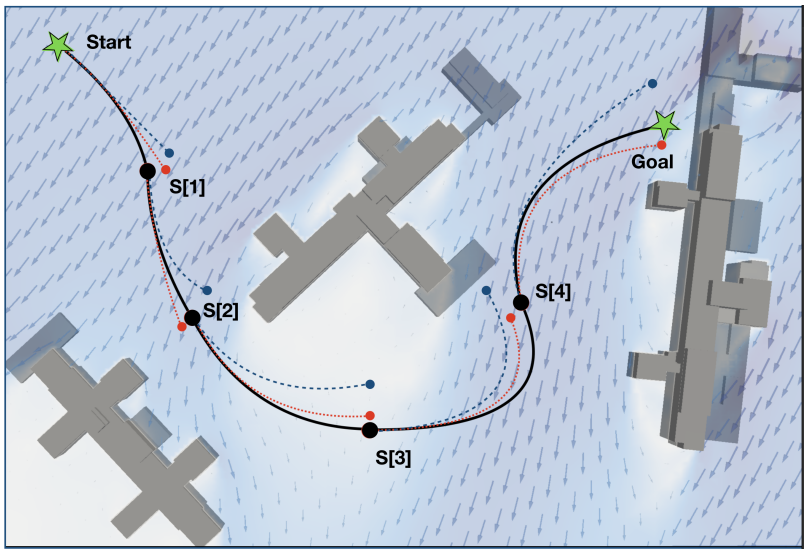


Figure 5.1: Figure shows a representative path planning scenario where blue lines indicate the BVP surrogate solutions chosen from the precomputed library, red indicates the wind-corrected solutions used for sampling-based planning algorithm (BIT*) and black lines represent the repaired final trajectory.

temporally static wind field. We present a BVP formulation that contains segments of higher-order polynomial spirals that can possess as many degrees of freedom as necessary to meet any number of constraints. For the offline phase, wind-agnostic solutions to pre-determined BVPs are used to populate a trajectory primitives library. These library primitives serve as surrogate solutions when the planner requests BVP solutions in the online phase. The requested BVP solutions are modified, assuming locally uniform wind. Once a solution is generated using these approximate primitives, we repair the path online using the actual nonlinear optimization BVP routine. As the repairs are only carried out on the final path, the number of actual BVPs solved is equal to the number of distinct edges in the final path. The final path is a wind-aware, smooth, collision-free sequence of states between a start and goal location. The approach is detailed in Figure 5.2

The main contributions outlined in this chapter are as follow:

- We formulate a novel wind-aware two-point boundary value problem that produces curvature, and curvature-rate constrained trajectories using piece-wise continuous curvature polynomials.

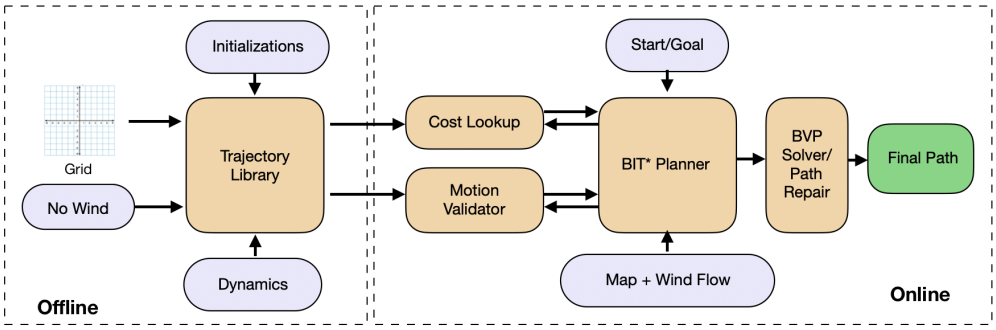


Figure 5.2: Overall Approach: Offline, we generate a trajectory library of precomputed wind-agnostic BVP solutions on a predefined grid. Online, we use the trajectory library to provide wind-aware surrogate solutions to perform real-time planning. Only the surrogate solutions that are part of the final path are repaired to provide smooth collision-free wind-aware path.

- We present an offline-online strategy using the surrogate-repair paradigm that uses a pre-computed trajectory library as surrogate solutions to provide real-time performance from a sampling-based planner.

5.2 Problem Definition

We define the UAV state space $\mathbb{C} = \mathbb{R}^3 \times \mathbb{S}^1$ where $\mathbf{x} \in \mathbb{C}$ is the state of the robot. If we define (x, y, z) as the position coordinates of the robot in inertial space and θ as the heading angle, then $\mathbf{x} = (x, y, z, \theta)^T$ is the vehicle state vector. Throughout the paper we assume a zero side-slip condition. Wind is assumed to be known and spatially-variant but temporally static in the horizontal x-y dimension with the vertical z component assumed to be zero. We further define $\mathbb{C}_{free}$ as the obstacle free state space of the robot. For this work, we use the vehicle kinematics model [54]

defined using trajectory distance variable $s \in [0, S_f]$ as follows:

$$x(s) = \int_0^s (cos\theta(s) + \frac{w_x}{V_a})ds \tag{5.1a}$$

$$y(s) = \int_0^s (sin\theta(s) + \frac{w_y}{V_a})ds \tag{5.1b}$$

$$z(s) = \int_0^s g(s)ds \tag{5.1c}$$

$$\theta(s) = \int_0^s \kappa(s)ds \tag{5.1d}$$

$$\tag{5.1e}$$

where V_a is the airspeed, w_x and w_y represent the x and y components of the wind vector in the inertial frame respectively[1], κ is the curvature and g is the glideslope. The input space is $\mathbf{u}(s) = (g(s), \kappa(s))^T$. Let $\sigma(s) = (\mathbf{x}(s), \mathbf{u}(s), S_f)^T$ define a path parameterized with $s \in [0, S_f]$. The choice of this non-holonomic system representation is used as it is consistent with many of the current UAV systems.

We may now formally define the problem. Given a start location $\mathbf{x}_s \in \mathbb{C}_{free}$ and a goal location $\mathbf{x}_g \in \mathbb{C}_{free}$, find the optimal path,

$$\begin{aligned} &\min_{\sigma(\cdot)} && S_f \\ &s.t. \quad \forall s \in [0, S_f] \\ &\mathbf{x}(s) \in \mathbb{C}_{free}, \;\; \mathbf{x}(0) = \mathbf{x}_s, \;\; \mathbf{x}(S_f) = \mathbf{x}_g \\ &|g(s)| \leq g_{max}, \;\; |\kappa(s)| \leq \kappa_{max}, \;\; |\dot{\kappa}(s)| \leq \dot{\kappa}_{max} \end{aligned} \tag{5.2}$$

5.3 Wind-Aware Boundary Value Problem

This section details our approach to formulate a BVP that gives a smooth, dynamically feasible wind-aware solution to exactly connect any states within $\mathbb{C}_{free}$. Traditionally, trochoids have been a popular candidate as a BVP solution for many of the wind-based path planners [1, 42, 43], because they intuitively represent the transformation due to wind. However, trochoids are discontinuous in curvature space which invalidates

[1]Here both w_x and w_y are functions of $\mathbf{x}$ and we use locally uniform values when solving the equations.

their use in our setup. We can still use the same idea and extend trochoids by using polynomials to represent the trochoidal curvature. Using a piece-wise continuous construction of these polynomials, it is possible to get curvature and curvature-rate limited paths that satisfy the boundary conditions. While the global planner is designed to deal with spatially nonuniform wind, for any two-point query we assume a local uniform wind profile by averaging the wind states between the two points. This approximation holds if the sampled states are spatially close assuming that the wind doesn't change drastically over relatively smaller distances (approx. 50m in our setup). The aforementioned sub-problem of finding a smooth trajectory between two states ignores any collision checks, as the global planner, BIT* in our case, handles these. Additionally, the only restriction on z-dimension is the maximum glideslope, and this is taken into consideration in the full planner. The BVP solution is generated in a SE2 space with $\tilde{\mathbf{x}} = (x, y, \theta)^T$ with κ as control input. This solution is then superimposed on a linear z-profile to connect the states in the actual $\mathbb{C}$ space.

5.3.1 BVP Formulation

Given an initial state $\tilde{\mathbf{x}}_0 = (x_0, y_0, \theta_0)^T$, a final state $\tilde{\mathbf{x}}_f = (x_f, y_f, \theta_f)^T$ and a uniform wind field $(w_x, w_y)^T$ between them, find a path $\Phi(s) = (\tilde{\mathbf{x}}(s), \kappa(s), sf)$ parameterized with $s \in [0, sf]$, where sf is path length such that it gives a $\Phi^*(\cdot)$

$$\begin{aligned} &\min_{\Phi(\cdot)} \quad sf \\ &s.t.\ \forall\ 0 \leq s \leq sf \\ &\kappa(s) \leq \kappa_{max}, \quad \kappa'(s) \leq \kappa'_{max}, \quad \kappa(0) = 0, \quad \kappa(sf) = 0 \\ &\tilde{\mathbf{x}}(0) = \tilde{\mathbf{x}}_\mathbf{0}, \qquad \tilde{\mathbf{x}}(sf) = \tilde{\mathbf{x}}_\mathbf{f} \end{aligned} \tag{5.3}$$

In order to solve this nonlinear optimization problem, we build on previous work [45] that uses polynomial spirals and extend it to generate three piece-wise continuous spirals of lengths sf_1, sf_2 and sf_3 to solve the problem. This formulation is based on Dubins formulation which uses a set of 3 curves with a bang-straight-bang strategy to connect two points [55] optimally. Such strategies have been shown to give optimal paths under the assumption that the distance between points is large enough. Thus the path Φ in Problem 5.3 can now be formulated as finding a continuous piece-wise

path $\Phi(\cdot)$ of length $sf = sf_1 + sf_2 + sf_3$,

$$\Phi(s) = \begin{cases} \phi_1(s) & 0 \leq s < sf_1 \\ \phi_2(s) & sf_1 \leq s < sf_1 + sf_2 \\ \phi_3(s) & sf_1 + sf_2 \leq s \leq sf \end{cases} \tag{5.4}$$

The i^{th} path where $i = \{1, 2, 3\}$ is defined as

$$\phi_i(s) = (\tilde{\mathbf{x}}(s), \kappa_i(s), sf_i)$$

where

$$\kappa_i(s) = a_i + b_i s + c_i s^2 + d_i s^3 + e_i s^4 \tag{5.5}$$

Keeping the curvature of the middle segment constant and simplifying using conditions in Eq. 5.3, we get

$$\kappa_1(s) = b_1 s + c_1 s^2 + d_1 s^3 + e_1 s^4 \tag{5.6a}$$
$$\kappa_2(s) = \kappa_1(sf_1) \tag{5.6b}$$
$$\kappa_3(s) = \kappa_2(sf_2) + b_3 s + c_3 s^2 + d_3 s^3 + e_3 s^4 \tag{5.6c}$$

These formulations give us the following set of 11 variables for the nonlinear optimization problem.

$$\mathbf{p} = [b_1, c_1, d_1, e_1, b_3, c_3, d_3, e_3, sf_1, sf_2, sf_3]$$

5.3.2 Nonlinear Optimization

The nonlinear optimization formulation is given as

$$\text{minimize:} \quad J(\mathbf{p}) = sf_1 + sf_2 + sf_3 \tag{5.7a}$$

such that,

$$||\Delta\mathbb{L}|| \leq \epsilon \tag{5.7b}$$

$$\text{for} \quad 0 \leq s < sf_1$$

$$|\kappa_1(s)| \leq \kappa_{max} \quad |\dot{\kappa}_1(s)| \leq \dot{\kappa}_{max} \tag{5.7c}$$

$$\text{for} \quad sf_1 \leq s < sf_1 + sf_2$$

$$|\kappa_2(s)| \leq \kappa_{max} \quad |\dot{\kappa}_2(s)| \leq \dot{\kappa}_{max} \tag{5.7d}$$

$$\text{for} \quad sf_1 + sf_2 \leq s \leq sf$$

$$|\kappa_3(s)| \leq \kappa_{max} \quad |\dot{\kappa}_3(s)| \leq \dot{\kappa}_{max} \tag{5.7e}$$

where,

$$\Delta\mathbb{L} = \begin{bmatrix} x(sf) - x_f \\ y(sf) - y_f \\ L(\theta(sf) - \theta_f) \\ L^2\kappa(sf) \end{bmatrix} \tag{5.8}$$

and ϵ is the error tolerance. The scaling factor L is used to weigh the angle and curvature error against positional error [45]. The terminal values are calculated using Equations 5.9.

$$x(sf) = \sum_{i=\{1,2,3\}} x_i(sf_i) + \frac{w_x}{V_a} \tag{5.9a}$$

$$y(sf) = \sum_{i=\{1,2,3\}} y_i(sf_i) + \frac{w_y}{V_a} \tag{5.9b}$$

$$\theta(sf) = \theta_3(sf_3) \tag{5.9c}$$

$$\kappa(sf) = \kappa_3(sf_3) \tag{5.9d}$$

where for i ={1,2,3},

$$\begin{aligned} x_i(sf_i) &= \int_0^{sf_i} cos(\theta_i(s))ds \\ y_i(sf_i) &= \int_0^{sf_i} sin(\theta_i(s))ds \\ \theta_i(s) &= \int_0^{s} \kappa_i(s)ds \end{aligned}$$

5.3.3 Initial guess

The nonlinear solver is sensitive to initial guess, and hence an informed initial guess that satisfies all the constraints is critical. Because we use 3 curvature-polynomials for the 3 segments, we may use a Dubins path to seed each of the 3 segments. To do this, we take each curve and first calculate the change of heading that each curve provides along with the direction of curvature (R or L). Using this, we solve a least-squares optimiser that outputs an initial vector based on the Dubins solution. Care is taken that the initial vector satisfies curvature and curvature rate constraints while formulating the least-squares problem. This process is only used to produce seeds for a subset of the paths after which solutions in the library may act as seeds to generate rest of the library.

5.4 BIT* Planner

This section details the domain-specific treatment of the BIT* path planning algorithm. BIT* is a class of tree-based incremental asymptotically optimal path planners. BIT* combines the use of heuristically ordered searches in a batch-wise fashion with the ability to focus the search on a sub-problem once the initial solution is available. These attributes significantly reduce the number of queries to find the actual cost function as opposed to RRT* making BIT* an attractive candidate for integration with BVPs [38, 39]. For a detailed explanation of the algorithm, we direct readers to the original paper[35].

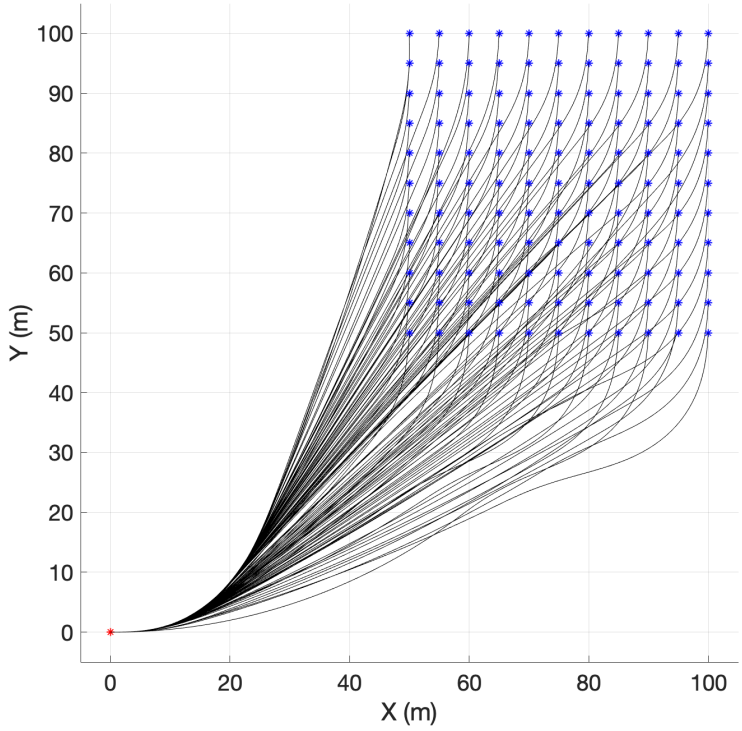


Figure 5.3: Representation of a section of the trajectory library with a 90° terminal angle

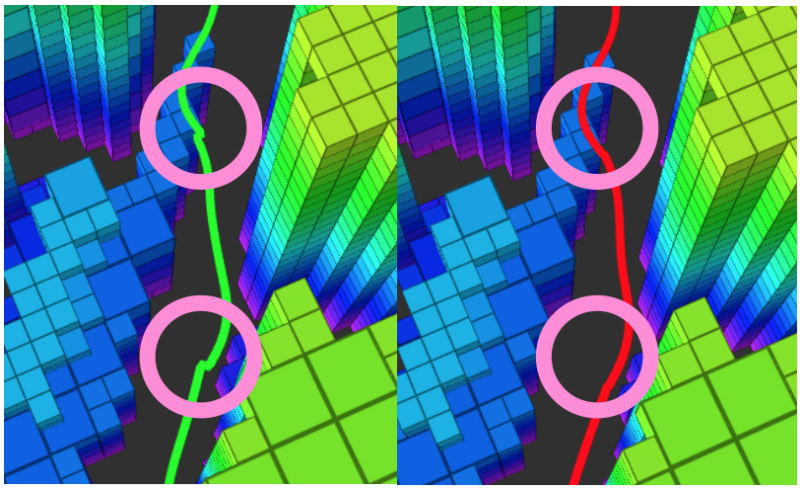


Figure 5.4: Paths before (left) and after (right) repairing the path

Algorithm 2 *GetPrimitive( )*

function GETPRIMITIVE($\mathbf{x}_s, \mathbf{x}_f, \mathbf{w}$)
  $[\mathbf{x_t}, \mathbf{w_t}] \leftarrow FwdTransform(\mathbf{x}_s, \mathbf{x}_f, \mathbf{w})$
  $c \leftarrow \infty$
  for all L $\in$ TrajLib **do** ▷ Loop over the library
    $[\mathbf{x}_L, \mathbf{p}_L] \leftarrow L$ ▷ Extract end point and primitive
    $\Delta\mathbf{x} \leftarrow \mathbf{x_L} - \mathbf{x_t}$ ▷ Spatial error
    $sf \leftarrow |\mathbf{p}_L(end-3:end)|_1$ ▷ Path length
    $c' \leftarrow |\Delta\mathbf{x} - \frac{\mathbf{w_t} sf}{V_a}|_2$ ▷ Wind-corrected error norm
    if $c' \leq c$ **then**
      $\mathbf{p} \leftarrow \mathbf{p}_L$
      $c \leftarrow c'$
    end if
  end for
  return p
end function

5.4.1 Trajectory Library in lieu of BVP computations

As mentioned earlier, BIT* relies on the BVP solution to give exact connections between the states. While the performance of our proposed solution to the BVP is in 100s of milliseconds (see Section 5.5), it is still not fast enough to be directly used to calculate cost functions and perform collision checks in real-time thus motivating the creation of a trajectory library containing pre-calculated primitives that can be used instead of solving the BVP. These solutions are modified in real-time using the wind conditions from the BIT* query. The trajectory library has precomputed solutions for a set of BVPs using a predetermined discretization of the state space. Without loss of generality, the start point of the BVP is kept at the origin, while the endpoint is varied inside the set bounds. Each solution containing the vector of optimization variable $\mathbf{p}$ is recorded along with the endpoint state. The BVPs are solved for a zero wind case. The level of discretization determines how close the actual solution will be from the query.

5.4.2 Algorithm Details

To successfully execute an instance of the planner, the planner needs the ability to query the cost and validity of a path between any two states $\{\mathbf{x}_s, \mathbf{x}_f\} \in \mathbb{C}_{free}$. For both these queries, it is important to find a primitive from the trajectory library that is spatially closest to the final path. A method to perform this search is captured in Algorithm 2. For every two-point query from the planner between states $\{\mathbf{x}_s, \mathbf{x}_f\}$ and the uniform wind $\mathbf{w}$ between them, we call the *GetPrimitive()* function. Using *FwdTransform()* [Line 2] the problem is first transformed into a frame of reference of the trajectory library. For this transformed problem, we iterate through the library [Line 4] to get the end point and the primitive for each member. The function then finds a member in the trajectory library [Line 5] which, when modified with the wind [Line 8], gives the spatially closest primitive to the transformed problem. The planner uses this surrogate solution to calculate the cost and perform collision checking.

The procedure for validity checking is detailed in Algorithm 3. To check if a path between two states is valid, i.e. $\in \mathbb{C}_{free}$, we first call *GetPrimitive()* [Line 2] to get the closest primitive. Then the spatial path is calculated from the primitive using *GetPath()* (refer Eq. 5.1 and 5.6). The path is then transformed back to the original frame using *BkdTransform()* [Line 4]. The function *CollisionAndGlideSlopeCheck()* [Line 5] validates if the states in the path are in free space, and also checks if the paths are within a glideslope threshold. When these checks pass, a path is returned as valid.

Once the planner returns a final path, we use the end-points of each section to construct the BVP and use the corresponding primitives as seeds to the nonlinear constrained optimization problem detailed in Equation 5.7 to give out the final solution. The repair process for creating a smooth continuous path can be seen in Figure 5.4.

5.5 Implementations and Feasibility Study

The BVP solver and global planner are implemented and tested in C++ to verify their real-time performance. In order to solve the nonlinear constrained optimization problem, we use COIN-OR's IPOPT solver [56]. We use the BIT* implementation in

Algorithm 3 *MotionValidator( )*

```
1: function MotionValidator(x_s, x_f, w)
2:     q ← GetPrimitive(x_s, x_f, w)
3:     Φ_t ← GetPath(q, w)
4:     Φ ← BkdTransform(Φ_t, x_s)
5:     if CollisionAndGlideSlopeCheck(Φ) then
6:         return Φ
7:     else
8:         return NULL
9:     end if
10: end function
```

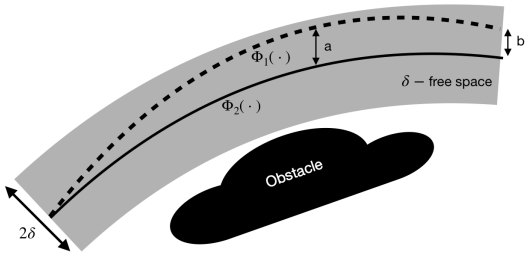


Figure 5.5: Representation of ϵ-ratio between any two paths. Figure also shows that path $\Phi_1(\cdot)$ is δ-close to path $\Phi_2(\cdot)$

OMPL [57] to generate a plan for every query. A custom validity checker and cost generator is implemented. The trajectory library is generated for a $\pm 600m$ range on both x and y with 5x5m resolution and a 10° resolution on θ.

In order to use the primitives as surrogates to the BVP solutions for the planner, we need to characterize how closely the primitives represent the actual solution. To do this, we find empirical bounds on metrics that quantify the difference between primitive and the corresponding repaired BVP solution.

5.5.1 Cost Metric

The cost metric is the percentage change in the cost between the seed and the solution. This metric quantifies how far from the actual cost is the surrogate cost used by the planner. In order to find this metric bound, we ran the BVP solver for 10,000 randomly sampled points within the bounds of the trajectory library with a varying wind of unit magnitude. The path lengths before and after repair were compared, and the percentage change is plotted in Figure 5.7 with a 95% confidence interval.

5.5.2 ϵ- metric

The ϵ- metric measures how much the path can spatially move after repairing. In order to define this metric, we need the following definitions.

Definition 1 *δ-clear state*

Any state $\mathbf{x} \in \mathbb{C}_{free}$ is said to be a δ-clear state if all the states within a δ euclidean distance from $\mathbf{x}$ are also in $\mathbb{C}_{free}$

Definition 2 *δ-clear path*

Any path $\Phi(\cdot)$ is said to be δ-clear path if all the states $\mathbf{x} \in \Phi(\cdot)$ are δ-clear states

Definition 3 *δ-clear space*

Any space within a δ euclidean distance of a δ-clear path $\Phi(\cdot)$ is said to be the δ-clear space of that $\Phi(\cdot)$ path.

Definition 4 *δ-close*

Given two paths $\Phi_1(\cdot)$ and $\Phi_2(\cdot)$, $\Phi_1(\cdot)$ is said to be δ-close to $\Phi_1(\cdot)$ if $\forall \mathbf{x} \in \Phi_1(\cdot)$, $\mathbf{x}$ are in δ-clear space of $\Phi_2(\cdot)$.

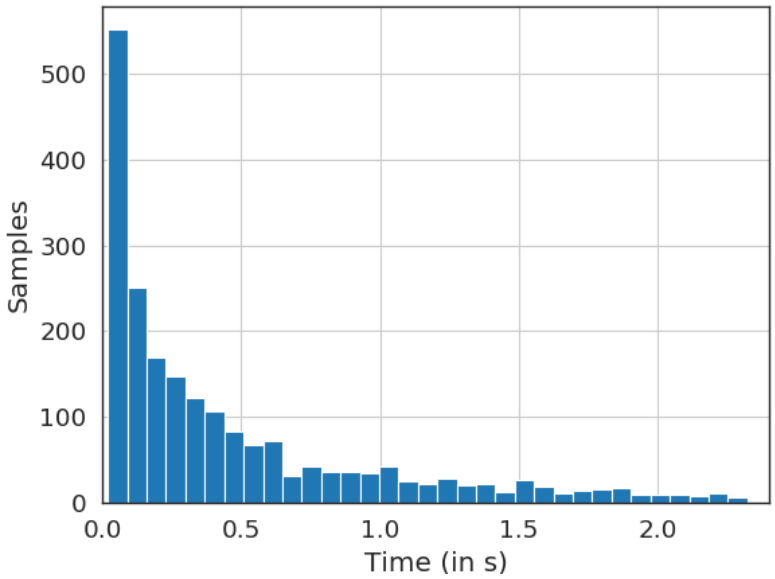


Figure 5.6: Histogram of time taken to solve the BVP based on 3000 samples drawn randomly from the state space.

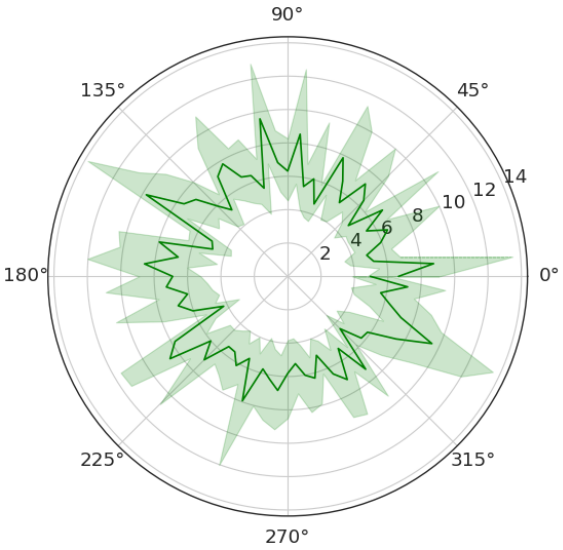


Figure 5.7: The percentage change in cost between the surrogate solution from the library and its corresponding repaired solution. Angles show the direction of incoming wind. As seen, the change hovers around a mean of 6.5%.

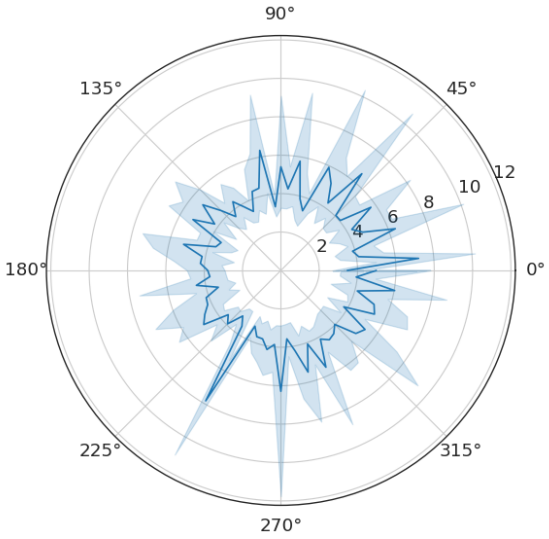
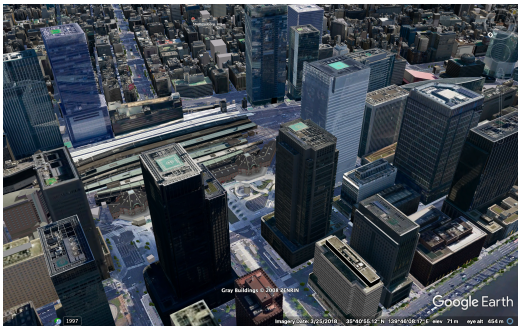
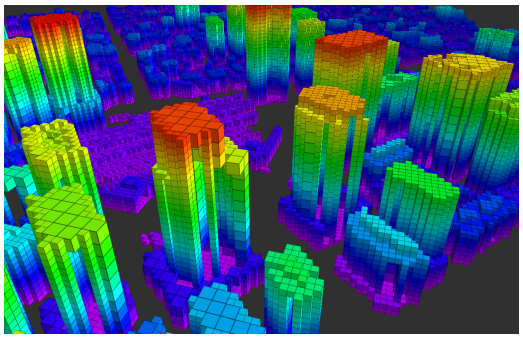


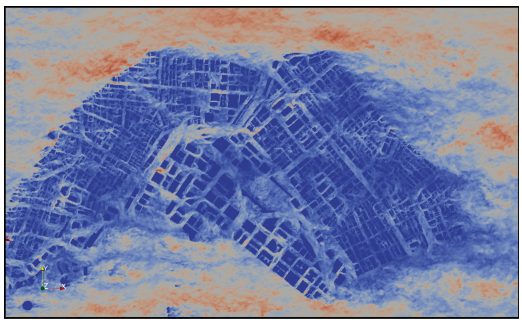
Figure 5.8: ϵ-ratio between the surrogate solution from the library and its corresponding repaired solution. Angles show the direction of incoming wind.



(a) 3D map of a city block



(b) Octomap representation of the city block used for planning



(c) Wind Field in the city

Figure 5.9: Urban environment used for simulation results

Definition 5 *ϵ-ratio*

Given two paths $\Phi_1(\cdot)$ and $\Phi_2(\cdot)$ with the same start states and path parametrization $t \in [0, \tau]$, an ϵ-ratio is defined:

$$\epsilon = \frac{max_{t\in[0,\tau]}(||\Phi_1(t) - \Phi_2(t)||)}{||\Phi_1(\tau) - \Phi_2(\tau)||}$$

Here we carry out a change of parametrization from s to t using $t(s) = \frac{s\tau}{sf}$ for both the paths.

Intuitively speaking, if we can put an empirical bound on the value of ϵ-ratio, then we can put a bound on how much a path is likely to diverge after it is repaired. This bound is a function of how far the endpoint of the surrogate is from the actual boundary condition, i.e. a solution is less likely to deviate from the surrogate if the boundary condition is almost satisfied. If the repaired path is δ-close to the surrogate solution, we can guarantee that the repaired path does not collide with any obstacles. Consider $\Phi_2(\cdot)$ in Figure 5.5 to be the selected surrogate solution and $\Phi_1(\cdot)$ be the corrected one. As the value of b is known, we may choose a δ inflation factor for the obstacles such that $\delta > \epsilon b$. For such a choice of δ, any repaired path will not collide with obstacles, and a collision check on the surrogate is equivalent to a collision check for the repaired path. In practice, getting a strict bound on the ϵ-ratio might be difficult, so we approximate the value by running multiple simulations and recording the value. In order to find the value, we ran the BVP solver in the same manner as for calculating the path length changes. The resulting value with a unit-magnitude wind direction and a 95% confidence interval is represented as radial plots in Figure 5.8.

5.5.3 Computation Time

Analysis of computation time required to find and repair a primitive is essential to prove the feasibility of the planner. Figure 5.6 shows a histogram of the time required to solve BVPs for 3000 randomly sampled points in the state space. A varying wind with unit magnitude is also applied. The mean time to solve a BVP is 484.6 milliseconds. The primitive lookup for the precomputed trajectory library

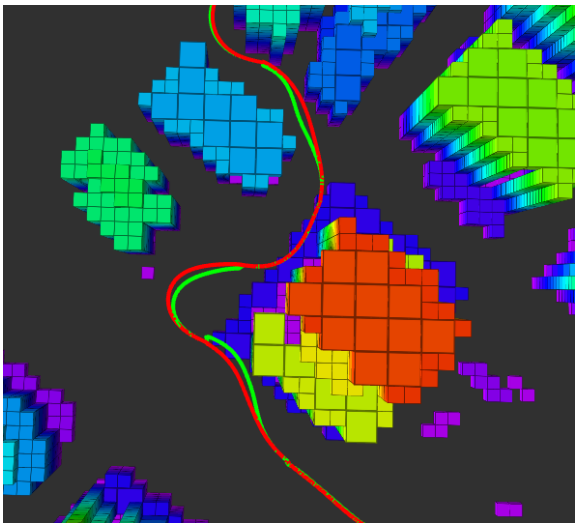


Figure 5.10: Figure shows a section of the calculated path in the simulation environment before (green) and after (red) repair

(Algorithm 1) can be slow if we iterate through the full library. One way to reduce this computation is to restrict the search in the trajectory library to an area which when transformed with wind is most likely to yield an optimal solution. Using this method we could achieve a lookup function that has a mean computation time of 0.256 milliseconds which is 3 orders of magnitude faster than the BVP computation time. All computations are performed on an Intel NUC8i5BEK with a processor base frequency of 2.30 GHz.

5.6 Numerical Results

In order to validate the efficacy of the proposed algorithms, we use a 3D model of an actual city block in a major metropolitan city. The city block is 3300 $\times$ 2130 meters and consists of several high rise structures. The maximum and minimum flight altitudes are restricted to 100m and 20m, respectively. Figures 5.9a and 5.9b show the actual city and its Octomap [58] representation with the same view. The 3D model of the city block is fed into a comprehensive high-fidelity CFD simulator to generate a temporally-static but spatially non-uniform wind field. The wind enters the city from the positive X direction with unit magnitude. The interactions between the wind and structures form interesting wind patterns, as shown in Figure 5.9c.

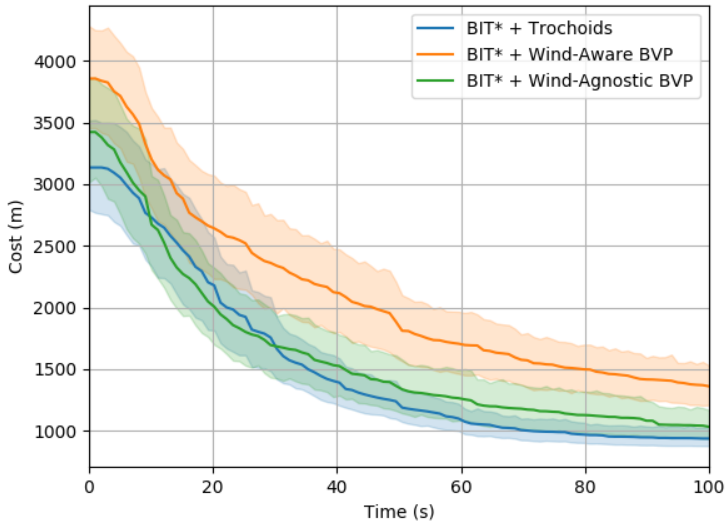


Figure 5.11: Figure shows the simulation results with the wind-aware and wind-agnostic BVP solver. Comparative results with trochoids show the competitive performance of our approach.

To verify the convergence properties of the planner, path planning queries are generated to construct trajectories from one location of the environment to the other. In order to compare the performance of the algorithm, we also present results using trochoids as a connector in the BIT*. Figure 5.10 shows results from one run. A run is successful if the algorithm can find a path using surrogate solutions, can repair all the segments in the path and the final repaired path is collision-free. The repairing routine is run with multiple initializations to improve BVP performance. Wind agnostic results are also shown for completeness. Two types of benchmark tests are performed: For the first set of runs, a start and goal location was selected, and the algorithms were tested with 100 different pseudo-random seeds for each. The trochoids based planner could successfully find a valid trajectory 98% of the time, while our proposed algorithm could find a feasible solution with a success rate of 96% for wind-agnostic and 84% for wind-aware cases. The path costs vs time for the test runs with 95% confidence bounds are shown in Figure 5.11. As trochoids can take curvature discontinuous turns, they generate lower-cost solutions. Our approach provides curvature continuous paths that are longer because they are smoother and

hence more feasible. Between wind-aware and wind-agnostic case, the costs are determined by the underlying flow field and in this specific case, the wind-aware costs are higher. While our approach does pay a penalty in success rate, the final costs and convergence rates are competitive against even curvature discontinuous trochoid based connectors. To further verify the performance of the planner, 100 sets of two points were randomly selected from $\mathbb{C}_{free}$ space, and path planning queries were generated. 93% of the queries were able to produce trajectories that were fully repaired and did not result in a collision.

5.7 Chapter Insights

A novel method for producing collision-free dynamically feasible trajectories in a known wind field is presented. Following are the major insights:

1. A BVP formulation using polynomial spirals to represent piece-wise continuous curvature is presented. The formulation uses a nonlinear optimization routine that respects constraints on curvature, curvature rate and glide-slope.
2. It was found that the optimization routine could not achieve competitive real-time performance and thus a method using a trajectory library of precomputed solutions that act as surrogates for the sampling based path planning algorithm is presented.
3. A feasibility study that defines and quantifies metrics to verify the applicability of the proposed method is undertaken. Empirically derived values for the metrics suggest that the method can strike a balance between speed and quality of solution.
4. Comparative results with trochoid BVPs in a similar setting shows the efficacy of the overall pipeline.

Chapter 6

Future Work

While we could address the problems discussed in the study to a degree, many challenges remain to realise the complete potential of UAVs flying low in cities. This chapter discuses the limitations and possible improvements of the proposed approaches.

For the wind estimation strategy, while the current flight test results show that the proposed methodology has potential to provide globally accurate wind field, a more comprehensive test routine will provide a clearer idea on the robustness of the algorithms to different atmospheric conditions and wind magnitudes. Targeted testing for estimating lower and higher magnitude wind will help better analyse the efficacy of algorithms. Our tests suggested that at lower wind magnitudes, the boundary value estimation errors are more pronounced. This may be attributed to two factors: a) the underlying CFD simulation loses its accuracy at lower wind speeds. b) the onboard wind measurement becomes less reliable at lower wind speeds. While the first issue can be addressed by using better high-fidelity CFD tools, the latter involves a deeper dive into characterizing the influence of propellers on the onboard anemometer. A wind tunnel test with the setup can provide valuable data on the relation between measured and actual wind speeds which may improve the overall efficiency of the algorithm.

For the wind planning setup, while the results are encouraging, more can be done to improve the robustness of the proposed methodology. Two drawbacks were identified. Firstly, while the use of surrogate solutions improves the real-time

performance of the sampling-based planner, it leads to a final sub-optimal solution. Secondly, there is no guarantee provided in the strategy that the chosen primitive will lead to convergence when attempting to repair the generated plan. Our tests suggested that the convergence rates for repairs are unfavourably affected by higher wind magnitudes. While a straightforward workaround is to try with multiple primitives while repairing, a more structured and possibly predictive approach can improve the system performance. Non-convergence may also result from the BVP trying to find solutions to dynamically unreachable states, as the planner has no information about the forward reachability of the state it is trying to expand. Most of these issues can be addressed by incorporating a more comprehensive strategy for choosing primitives from the trajectory library. One possible approach is to assign a probability distribution on the trajectory library conditioned on the query at hand. Such a distribution can encode how likely is a particular primitive to converge on a given query. One other drawback of the current work is the lack of field test results with a real multirotor UAV. These will be added in subsequent work.

One other possible direction of work would be to develop more informative objective functions. The proposed methodology uses minimizing path length as an objective. While following the shortest length path seems like a plausible objective given a constant ground-speed, the effects of wind would lead us to the fact that a shorter path objective might not translate well as meaningful metric in the real windy world. A time-based metric is the next possible candidate. A UAV that can tap into the underlying flow-field can reach its destination faster by avoiding areas of headwinds and actively looking for tailwind. So even though a path might be spatially longer, the presence of an underlying flow-field necessitates the use of an objective function that generates a shorter time path. Taking this a step further, we may want to construct an energy-based metric. With a UAV operating at a constant power, the time and energy based metrics will align. While an energy-based metric might be more accurate in some cases (eg. crosswinds), it is also useful in predicting the overall energy consumption. Given the relatively short battery capacities, optimizing directly for energy consumption and providing an estimate of the energy usage for following a path can provide the system with critical information that dictates the operational envelope and the safety characteristics of UAVs.

Overall the complete estimation-planning strategy still needs to be verified end-

to-end in practice. While both the blocks by themselves are tested in this work, studying the interaction between them while running a complete dummy mission and investigating their ability to support each other would be an interesting direction for future research.

Chapter 7

Conclusions

The aim of this thesis is to identify the challenges of operating UAVs at low-altitude dense urban environments and propose methods to address these challenges. The focus is on UAV operations in windy conditions. This thesis identifies two research gaps that need to be addressed to overcome the challenges.

The first is the lack of a comprehensive approach to estimate the complex underlying flow field in a urban setup which may enable wind-aware planning over longer horizons. A method to use onboard wind measurements from a UAV to solve the inverse problem of estimating the inlet conditions to improve CFD-based wind field estimation is presented. The work shows that it is possible to reliably use an anemometer on a moving UAV platform as sources of local wind measurements. The measurements are then used in a particle filter to provide a posterior distribution of the inlet conditions. A Gaussian Process Regression model is used as a surrogate function to achieve real-time performance. The method is implemented on a multirotor platform and the results are shown to be consistent with the measurements from a static roof-mounted weather station.

The second gap is the lack of a planning solution that finds curvature continuous and wind-aware paths in cluttered spaces in real time. While there is some work in finding curvature continuous and wind-aware paths, no previous work has a unified approach to address both these issues. A novel method for producing collision-free dynamically feasible trajectories in a known wind field is presented. To the best of authors' knowledge, this is the only real-time near-optimal BVP-based implementation

for curvature continuous wind-aware path planning. The proposal uses a library of precomputed two-point boundary value problem solutions as connectors in sampling-based planners. The final path was produced by repairing the surrogate solutions to produce a smooth continuous path. Benchmark tests in a realistic simulation environment are performed, and comparative results with a trochoid-based BVP planner are used to showcase the performance of the proposed strategy. The algorithm was able to produce wind-aware curvature-continuous paths in competitive time frames and cost ranges.

Bibliography

[1] L. Techy and C. A. Woolsey, "Minimum-time path planning for unmanned aerial vehicles in steady uniform winds," *Journal of guidance, control, and dynamics*, vol. 32, no. 6, pp. 1736–1746, 2009. (document), 3.1, 3.2, 5.3

[2] E. Demaitre, "Covid-19 pandemic prompts more robot usage worldwide," Mar 2020. 1.1

[3] S. Hasija and A. Gumaledar, "Are robots overrated?," Apr 2020. 1.1

[4] "68% of the world population projected to live in urban areas by 2050, says un — un desa department of economic and social affairs." 1.1

[5] B. Z. Cybyk, B. E. McGrath, T. M. Frey, D. G. Drewry, J. F. Keane, and G. Patnaik, "Unsteady airflows and their impact on small unmanned air systems in urban environments," *Journal of Aerospace Information Systems*, vol. 11, no. 4, pp. 178–194, 2014. 1.1, 4.1

[6] S. M. LaValle and J. J. Kuffner Jr, "Randomized kinodynamic planning," *The international journal of robotics research*, vol. 20, no. 5, pp. 378–400, 2001. 2.1.2

[7] L. E. Dubins, "On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents," *American Journal of mathematics*, vol. 79, no. 3, pp. 497–516, 1957. 2.1.2

[8] D. Mellinger and V. Kumar, "Minimum snap trajectory generation and control for quadrotors," in *2011 IEEE international conference on robotics and automation*, pp. 2520–2525, IEEE, 2011. 2.1.2

[9] T. Niedzielski, C. Skjøth, M. Werner, W. Spallek, M. Witek, T. Sawiński, A. Drzeniecka-Osiadacz, M. Korzystka-Muskała, P. Muskała, P. Modzel, *et al.*, "Are estimates of wind characteristics based on measurements with pitot tubes and gnss receivers mounted on consumer-grade unmanned aerial vehicles applicable in meteorological studies?," *Environmental monitoring and assessment*, vol. 189, no. 9, p. 431, 2017. 3.1

[10] M. Marino, A. Fisher, R. Clothier, S. Watkins, S. Prudden, and C. S. Leung, "An evaluation of multi-rotor unmanned aircraft as flying wind sensors," *International*

Journal of Micro Air Vehicles, vol. 7, no. 3, pp. 285–299, 2015. 3.1

[11] N. Lawrance and S. Sukkarieh, "Simultaneous exploration and exploitation of a wind field for a small gliding uav," in *AIAA Guidance, Navigation, and Control Conference*, p. 8032, 2010. 3.1

[12] J. W. Langelaan, J. Spletzer, C. Montella, and J. Grenestedt, "Wind field estimation for autonomous dynamic soaring," in *2012 IEEE International conference on robotics and automation*, pp. 16–22, IEEE, 2012. 3.1

[13] L. Rodriguez, J. A. Cobano, and A. Ollero, "Wind field estimation and identification having shear wind and discrete gusts features with a small uas," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 5638–5644, IEEE, 2016. 3.1

[14] M. W. Orr, S. J. Rasmussen, E. D. Karni, and W. B. Blake, "Framework for developing and evaluating mav control algorithms in a realistic urban setting," in *Proceedings of the 2005, American Control Conference, 2005.*, pp. 4096–4101, IEEE, 2005. 3.1, 4.1

[15] D. Galway, J. Etele, and G. Fusina, "Modeling of urban wind field effects on unmanned rotorcraft flight," *Journal of aircraft*, vol. 48, no. 5, pp. 1613–1620, 2011. 3.1, 4.1

[16] J. Ware and N. Roy, "An analysis of wind field estimation and exploitation for quadrotor flight in the urban canopy layer," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1507–1514, IEEE, 2016. 3.1, 3.2, 4.1

[17] A. Ricci, I. Kalkman, B. Blocken, M. Burlando, and M. Repetto, "Impact of turbulence models and roughness height in 3d steady rans simulations of wind flow in an urban environment," *Building and Environment*, vol. 171, p. 106617, 2020. 3.1

[18] C. García-Sánchez, D. Philips, and C. Gorlé, "Quantifying inflow uncertainties for cfd simulations of the flow in downtown oklahoma city," *Building and environment*, vol. 78, pp. 118–129, 2014. 3.1

[19] D. Wise, V. Boppana, K. Li, and H. Poh, "Effects of minor changes in the mean inlet wind direction on urban flow simulations," *Sustainable cities and society*, vol. 37, pp. 492–500, 2018. 3.1

[20] J. Sousa, C. García-Sánchez, and C. Gorlé, "Improving urban flow predictions through data assimilation," *Building and Environment*, vol. 132, pp. 282–290, 2018. 3.1, 4.2.3

[21] J. Sousa and C. Gorlé, "Computational urban flow predictions with bayesian inference: Validation with field data," *Building and Environment*, vol. 154,

pp. 13–22, 2019. 3.1

[22] S. Datta Gupta, "A comparative study of the particle filter and the ensemble kalman filter," Master's thesis, University of Waterloo, 2009. 3.1

[23] A. S. Stordal, H. A. Karlsen, G. Nævdal, H. J. Skaug, and B. Vallès, "Bridging the ensemble kalman filter and particle filters: the adaptive gaussian mixture filter," *Computational Geosciences*, vol. 15, no. 2, pp. 293–305, 2011. 3.1

[24] C. Liu, O. McAree, and W.-H. Chen, "Path following for small uavs in the presence of wind disturbance," in *Proceedings of 2012 UKACC International Conference on Control*, pp. 613–618, IEEE, 2012. 3.2

[25] A. Ailon and I. Zohar, "Control of unmanned aerial vehicle with restricted input in the presence of additive wind perturbations," in *2011 11th International Conference on Control, Automation and Systems*, pp. 873–878, IEEE, 2011. 3.2

[26] J. Patrikar, V. R. Makkapati, A. Pattanaik, H. Parwana, and M. Kothari, "Nested saturation based guidance law for unmanned aerial vehicles," *Journal of Dynamic Systems, Measurement, and Control*, vol. 141, no. 7, p. 071008, 2019. 3.2

[27] S. L. Herbert, M. Chen, S. Han, S. Bansal, J. F. Fisac, and C. J. Tomlin, "Fastrack: a modular framework for fast and guaranteed safe motion planning," in *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, pp. 1517–1522, IEEE, 2017. 3.2

[28] V. Stepanyan and K. S. Krishnakumar, "Estimation, navigation and control of multi-rotor drones in an urban wind field," in *AIAA Information Systems-AIAA Infotech@ Aerospace*, p. 0670, 2017. 3.2

[29] D. N. Subramani, Q. J. Wei, and P. F. Lermusiaux, "Stochastic time-optimal path-planning in uncertain, strong, and dynamic flows," *Computer Methods in Applied Mechanics and Engineering*, vol. 333, pp. 218–237, 2018. 3.2

[30] I. Mir, S. A. Eisa, and A. Maqsood, "Review of dynamic soaring: technical aspects, nonlinear modeling perspectives and future directions," *Nonlinear Dynamics*, vol. 94, no. 4, pp. 3117–3144, 2018. 3.2

[31] N. R. Lawrance and S. Sukkarieh, "Path planning for autonomous soaring flight in dynamic wind fields," in *2011 IEEE international conference on robotics and automation*, pp. 2499–2505, IEEE, 2011. 3.2

[32] A. Bahabry, X. Wan, H. Ghazzai, H. Menouar, G. Vesonder, and Y. Massoud, "Low-altitude navigation for multi-rotor drones in urban areas," *IEEE Access*, vol. 7, pp. 87716–87731, 2019. 3.2

[33] S. Primatesta, G. Guglieri, and A. Rizzo, "A risk-aware path planning strategy for uavs in urban environments," *Journal of Intelligent & Robotic Systems*,

vol. 95, no. 2, pp. 629–643, 2019. 3.2

[34] B. Nurimbetov, O. Adiyatov, S. Yeleu, and H. A. Varol, "Motion planning for hybrid uavs in dense urban environments," in *2017 IEEE International Conference on Advanced Intelligent Mechatronics (AIM)*, pp. 1627–1632, IEEE, 2017. 3.2

[35] J. D. Gammell, S. S. Srinivasa, and T. D. Barfoot, "Batch informed trees (bit*): Sampling-based optimal planning via the heuristically guided search of implicit random geometric graphs," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3067–3074, IEEE, 2015. 3.2, 5.4

[36] S. Karaman, M. R. Walter, A. Perez, E. Frazzoli, and S. Teller, "Anytime motion planning using the rrt," in *2011 IEEE International Conference on Robotics and Automation*, pp. 1478–1483, IEEE, 2011. 3.2

[37] R. E. Allen and M. Pavone, "A real-time framework for kinodynamic planning in dynamic environments with application to quadrotor obstacle avoidance," *Robotics and Autonomous Systems*, vol. 115, pp. 174–193, 2019. 3.2

[38] C. Xie, J. van den Berg, S. Patil, and P. Abbeel, "Toward asymptotically optimal motion planning for kinodynamic systems using a two-point boundary value problem solver," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 4187–4194, IEEE, 2015. 3.2, 5.4

[39] M. Lan, S. Lai, Y. Bi, H. Qin, J. Li, F. Lin, and B. M. Chen, "Bit*-based path planning for micro aerial vehicles," in *IECON 2016-42nd Annual Conference of the IEEE Industrial Electronics Society*, pp. 6079–6084, IEEE, 2016. 3.2, 5.4

[40] B. Sakcak, L. Bascetta, G. Ferretti, and M. Prandini, "Sampling-based optimal kinodynamic planning with motion primitives," *Autonomous Robots*, pp. 1–18, 2019. 3.2

[41] Y. Li, Z. Littlefield, and K. E. Bekris, "Sparse methods for efficient asymptotically optimal kinodynamic planning," in *Algorithmic foundations of robotics XI*, pp. 263–282, Springer, 2015. 3.2

[42] S. Schopferer and T. Pfeifer, "Performance-aware flight path planning for unmanned aircraft in uniform wind fields," in *2015 International Conference on Unmanned Aircraft Systems (ICUAS)*, pp. 1138–1147, IEEE, 2015. 3.2, 5.3

[43] S. Schopferer, J. S. Lorenz, A. Keipour, and S. Scherer, "Path planning for unmanned fixed-wing aircraft in uncertain wind conditions using trochoids," in *2018 International Conference on Unmanned Aircraft Systems (ICUAS)*, pp. 503–512, IEEE, 2018. 3.2, 5.3

[44] M. Seleckỳ, P. Váňa, M. Rollo, and T. Meiser, "Wind corrections in flight path planning," *International Journal of Advanced Robotic Systems*, vol. 10, no. 5,

p. 248, 2013. 3.2

[45] A. Kelly and B. Nagy, "Reactive nonholonomic trajectory generation via parametric optimal control," *The International Journal of Robotics Research*, vol. 22, no. 7-8, pp. 583–601, 2003. 3.2, 5.3.1, 5.3.2

[46] M. Kothari, I. Postlethwaite, and D.-W. Gu, "Uav path following in windy urban environments," *Journal of Intelligent & Robotic Systems*, vol. 74, no. 3-4, pp. 1013–1028, 2014. 4.1

[47] S. Nakano, G. Ueno, and T. Higuchi, "Merging particle filter for sequential data assimilation," 2007. 4.1

[48] S. Thrun, W. Burgard, and D. Fox, *Probabilistic robotics*, vol. 1. MIT press Cambridge, 2000. 4.2.1, 4.2.2

[49] O. Garibaldi Castillo and E. Londner, "Uav flight testing with an airborne sonic anemometer, imu and gps," in *28th AIAA Applied Aerodynamics Conference*, p. 4572, 2010. 4.2.4

[50] P. Bruschi, M. Piotto, F. Dell'Agnello, J. Ware, and N. Roy, "Wind speed and direction detection by means of solid-state anemometers embedded on small quadcopters," *Procedia Engineering*, vol. 168, pp. 802–805, 2016. 4.2.4

[51] R. L. Thorpe, M. McCrink, and J. W. Gregory, "Measurement of unsteady gusts in an urban wind field using a uav-based anemometer," in *2018 Applied Aerodynamics Conference*, p. 4218, 2018. 4.2.4

[52] "OpenFOAM," 2020. 4.3.2

[53] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011. 4.3.2

[54] V. Dugar, S. Choudhury, and S. Scherer, "A κite in the wind: Smooth trajectory optimization in a moving reference frame," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 109–116, IEEE, 2017. 5.2

[55] Y. Lin and S. Saripalli, "Path planning using 3d dubins curve for unmanned aerial vehicles," in *2014 international conference on unmanned aircraft systems (ICUAS)*, pp. 296–304, IEEE, 2014. 5.3.1

[56] A. Wächter and L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Mathematical programming*, vol. 106, no. 1, pp. 25–57, 2006. 5.5

[57] I. A. Şucan, M. Moll, and L. E. Kavraki, "The Open Motion Planning Library," *IEEE Robotics & Automation Magazine*, vol. 19, pp. 72–82, December 2012.

http://ompl.kavrakilab.org. 5.5

[58] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "Octomap: An efficient probabilistic 3d mapping framework based on octrees," *Autonomous robots*, vol. 34, no. 3, pp. 189–206, 2013. 5.6