Flying Like a Pilot in Wind: Smooth Trajectory Optimization in a Moving Reference Frame

Vishal Dugar

CMU-RI-TR-17-50

July 2017

Robotics Institute
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

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

Keywords: Autonomous aerial vehicles, motion planning, trajectory optimization, wind

Abstract

A significant challenge for unmanned aerial vehicles capable of flying long distances is planning in a wind field. Although there has been a plethora of work on the individual topics of planning long routes, smooth trajectory optimization and planning in a wind field, it is difficult for these methods to scale to solve the combined problem. In this thesis, we address the problem of planning long, dynamically feasible, time-optimal trajectories in the presence of wind (which creates a moving reference frame). Additionally, we attempt to solve for trajectories that exhibit features similar to the flight profiles of human pilots.

We present an algorithm, κITE , that elegantly decouples the joint trajectory optimization problem into individual path optimization in a fixed ground frame and a velocity profile optimization in a moving reference frame. The key idea is to derive a decoupling framework that guarantees feasibility of the final fused trajectory. Our results show that κITE produces high-quality solutions for planning with a full-size helicopter flying at speeds of 50 m/s, handling winds up to 20 m/s and missions over 200 km. We validate our approach with real-world experiments on a full-scale helicopter with a pilot in the loop. Our approach paves the way forward for autonomous systems to exhibit pilot-like behavior when flying missions in winds aloft.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Figure 1.1: A modified Bell 206 serves as one of our autonomous helicopter test platforms.

There has recently been extensive research on unmanned aerial vehicles (UAVs) such as helicopters and fixed-wing aircraft that can travel large distances [1, 9, 15, 27, 37]. The commercial success of such systems depends heavily on their ability to produce high-performance flight profiles that optimize time while strictly adhering to constraints imposed by the control system, flight dynamics and performance charts [12, 29]. In conjunction with these requirements, these systems must be cognizant of the effect of wind on flight profiles [12, 26, 33, 34]. We therefore address the problem of planning time-optimal trajectories that are dynamically feasible in a moving reference frame, and remain in a specified safe flight corridor.

Consider the problems faced by the autonomous helicopter shown in Fig. 1.1 when planning in a moving reference frame. Planning a dynamically feasible path in this frame results in a drifting ground frame path that might violate the safe flight corridor. On the other hand, if

Figure 1.2: A helicopter crash in Fiji. A strong gust of wind destabilized it as it attempted to land.

planning is done in the ground frame, the dynamics constraints are no longer stationary and vary along the path. Failing to account for wind during planning can lead to trajectories that exceed control margins, which can result in a catastrophic failure - an example is shown in Fig. 1.2, where a manned helicopter crashed in Fiji due to gusty winds. In addition to wind, the other main challenges are satisfying non-holonomic constraints due to vehicle dynamics and scaling to large distances - these systems must often operate over mission lengths exceeding $200km$. This results in a complex multi-resolution, non-convex planning problem. It is also important for the solution to resemble the flight profiles of human pilots, especially in cases where these unmanned systems carry human passengers. Finally, the optimizer is required to have near real-time behavior. The need to re-plan a trajectory potentially extending over hundreds of kilometers online (within a minute or two) can be caused due to change in wind conditions or high-level mission requirements.

We present an algorithm, κITE (Curvature (κ) parameterization Is very Time Efficient) [10] [11], to efficiently solve this optimization problem. We summarize the key ideas behind the effectiveness of the algorithm as follows –

1. We decouple the trajectory optimization problem into a path optimization problem in ground frame and velocity optimization in airframe. This decoupling is done in such a way so as to ensure that when the individual outputs are fused together, the trajectory is guaranteed to be feasible.
2. We use an efficient piecewise curvature polynomial parameterization to solve the path optimization problem that can scale with distance and waypoints.

3. We use a two-step velocity profile optimizer to solve efficiently for a coarse profile and subsequently refine it with piecewise velocity polynomials.

1.1 Notation

We make use of the following notation throughout this text.

Table 1.1: Notation

Symbol	Description	Symbol	Description
x	x-coord in air frame	x_g	x-coord in ground frame
y	y-coord in air frame	y_g	y-coord in ground frame
ϕ	roll in air frame	ϕ_g	roll in ground frame
ψ	yaw in air frame	ψ_g	yaw in ground frame
v	airspeed	v_g	groundspeed
a	acc in air frame	a_g	acc in ground frame
j	jerk in air frame	j_g	jerk in ground frame
σ	traj in air frame	σ_g	traj in ground frame
ξ	path in air frame	ξ_g	path in ground frame
s	arc distance	τ	index

Chapter 2

Background

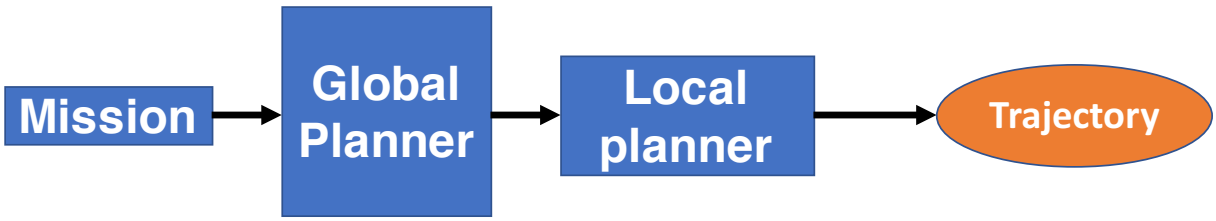


Figure 2.1: The motion planning architecture.

2.1 Architecture

Our work falls into the realm of generating kinodynamic trajectories with non-holonomic motion constraints [22], [21]. The motion planning architecture that we use is shown in Fig. 2.1, where a mission representing an initial *obstacle-free* configuration of waypoints and safe corridors is passed to a global planner. The global planner outputs a complete (from start to goal), dynamically feasible trajectory lying within the safe corridors, and a local planner attempts to follow this trajectory up to a horizon while avoiding obstacles that may suddenly appear. The focus of this work is on the global planner.

Our choice of architecture, combining a global planner with a local planner, is driven by the need to reason about long horizons. Large aerial vehicles such as helicopters and fixed-wing aircraft often fly very long distances, and it is important to have some feasibility guarantees on the entire trajectory even before the aircraft takes off. For instance, let us consider the helicopter shown in Fig. 4.1. This helicopter must fly from start to goal in the presence of wind, while remaining within the safe flight corridor at all times. If the trajectory planner were to plan to a limited horizon (say, until the first turn), there would be no guarantee that the helicopter would be able to feasibly execute the second turn given its state at the end of the first turn. Similarly, changing wind conditions might make landing at the touchdown point completely infeasible. It is therefore important for the trajectory planner to reason about the entire route, and not just a portion of it. The global planner can also satisfy higher-level requirements such as pilot-like behavior. This global planner feeds into a local planner (say, a sampling-based planner) that

attempts to follow this trajectory while avoiding obstacles. This allows us to *efficiently* decouple the problem of reasoning globally about the entire route from local requirements such as avoiding sudden obstacles, and works well in our experience from real-world flight tests. Expecting a single planner to reason both about global feasibility and higher-order (e.g. pilot-like flight) requirements, as well as avoiding obstacles locally, proves to be intractable to solve in quick time, especially for long missions.

2.2 Related Work

Nonholonomic trajectory generation is a hard problem, and can generally only be solved approximately using prior assumptions that depend on the environment and dynamics. Leveraging intelligent prior assumptions, however, allows us to find high-quality (albeit suboptimal) solutions in quick time. One of these priors relates to the structure of the solution, especially for curvature-constrained systems. Continuous curvature systems have previously been tackled by [13], [31]. In this thesis, we build on the use of curvature polynomials in [20] for parameterizing turns. Apart from this, local optimization techniques such as [30], [18], [32] have also been used for generating trajectories, especially in the context of manipulators. State lattice techniques [28], [23], [25] have found success in planning trajectories for mobile robots, but they require careful tuning in terms of the resolution, lattice design and the search algorithm.

The problem of generating feasible trajectories for UAVs has previously been explored in the literature. [2] builds on the Dubins solution and uses fixed-radius arcs to link straight segments with constant speed. [17] proposes an online, corridor-constrained smoothing algorithm that uses B-spline templates to generate paths, but not time profiles. Sampling-based techniques like [14], [19] are quite popular, but do not scale well with problem size. There has also been some work that deals with trajectory optimization in the presence of wind. The classic Zermelo–Markov–Dubins problem has been studied in [3], [34] and [4] to characterize optimal solutions, but they use sharp turns and constant speed, neither of which are practical. [26] also uses a bounded turning radius assumption to yield minimum-time trajectories with constant speed. [35] uses a bounded roll-rate to construct smooth, continuous-curvature paths between two states in the presence of wind. However, it again assumes constant speed and does not provide a mechanism to extend the method to variable-speed trajectories. Tables 2.1 and 2.2 summarize the different approaches to this problem.

Since practical trajectory planning problems are extremely hard, it is often necessary to decouple the problem into an initial path-finding stage and a subsequent velocity-optimization process ([6], [5], [16]). We use concepts from previous work done on optimizing velocity profiles given a fixed path and a finite set of velocity bottlenecks ([24], [36]) to compute time-optimal velocity profiles.

Table 2.1: Comparison of planning approaches

Planning approach	**Feasibility**	**Time optimality**	**Long-horizon reasoning**	**Time efficiency**	**Encode pilot-like behavior**
κITE (our approach)	Guaranteed to be dynamically and spatially feasible	Sub-optimal in time	Reasons about complete trajectory	Near real-time	Easy to encode pilot-like behavior
Sampling-based techniques (e.g. [14], [19], [16])	Non-trivial to compute feasible connect functions in wind	Sub-optimal in finite time	Can reason about complete trajectory, but very difficult to scale to large horizons in quick time	Efficient for smaller horizons, but large search time for longer horizons	Non-trivial to encode pilot-like behavior
Lattice-based techniques (e.g. [28], [23], [25])	Non-trivial to compute feasible lattice edges in wind	Sub-optimal in the anytime setting; optimal if given enough time, assuming sufficient lattice resolution	Can reason about complete trajectory, but very difficult to scale to large horizons in quick time	Efficient for small horizons, but large search time for longer horizons	Non-trivial to encode pilot-like behavior
Trajectory optimization approaches (e.g. [30], [18], [32])	Non-trivial to guarantee feasibility in wind	Locally optimal with a good initialization. One approach could be to use κITE as the initializer.	Can reason about complete trajectory	Can be efficient with good initialization	Non-trivial to encode pilot-like behavior

Table 2.2: Comparison of planning approaches (continued)

Planning approach	**Feasibility**	**Time optimality**	**Long-horizon reasoning**	**Time efficiency**	**Encode pilot-like behavior**
UAV planning in wind (e.g. [4], [26], [35])	Feasible under a constant-speed assumption	Optimal turns under a constant speed assumption	Constant speed assumption does not work for planning complete missions	Turns can be efficiently generated	Non-trivial to emulate pilot-like behavior, especially with constant speed and sharp turning-radius assumptions
Using a controller/filter on an initial guess (e.g. [8])	Feasible over short horizons, assuming controller accounts for wind	Sub-optimal in time	Non-trivial to guarantee feasibility over long horizons	Quick to compute	Can emulate pilot-like behavior if it can be encoded in the controller

Chapter 3

Flying Like a Pilot

Consider a human helicopter pilot - he or she optimizes an unknown cost function while flying, balancing time efficiency and smoothness while attempting to remain safe at all times. While it is impossible to pin down this cost function exactly, we have obtained a fair idea through our conversations with test pilots, examining flight logs and by studying the aviation literature [12].

3.1 Frames of Reference

Aerial vehicles fly in air, which is often moving with respect to the ground (wind). Similar to moving across a river, this moving air introduces a new, moving frame of reference that must be dealt with. Thus, planning a helicopter's trajectory must occur over the two frames shown in Fig. 3.1–

- An earth-fixed ground frame, $\mathcal{G}$, and
- an air-frame, $\mathcal{A}$

For the sake of simplicity, let us assume that wind is directed along the +ve x-axis. In this situation, the speed of the vehicle in the two systems is related by the following-

$$v_g = v + v_w \tag{3.1}$$

The ground-referenced velocity is therefore the vector summation of the air-referenced velocity and the wind velocity (measured with respect to the ground frame). While a lot of the exposition here will assume a constant wind directed along the x-axis, all the methods can trivially be extended to the more realistic scenario where wind varies along the length of the trajectory. The implementation of the algorithm that runs on our test platform relies on a wind estimator function that returns estimates of the wind speed and direction along the trajectory. The only constraint placed on the wind estimate is that it should have a finite third derivative, since the final trajectory is required to be bounded in jerk (the third derivative) as well.

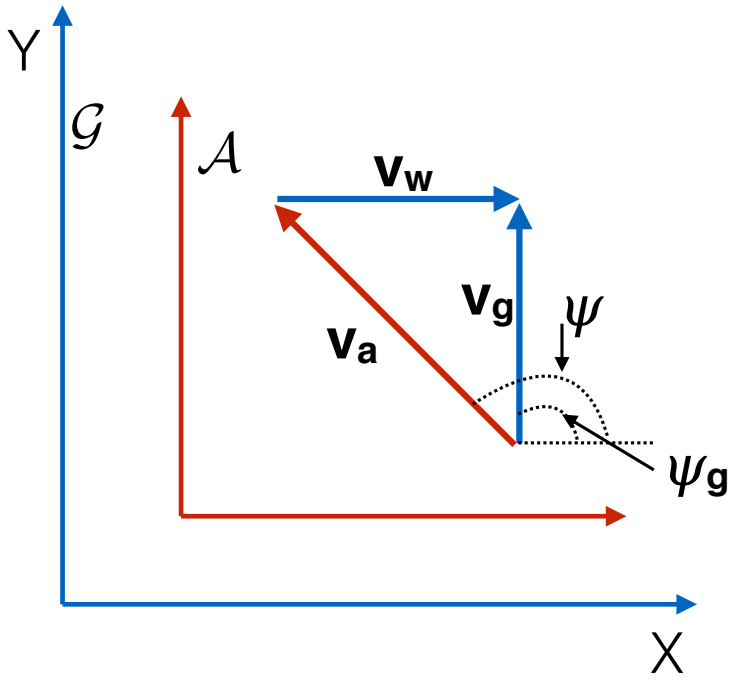


Figure 3.1: Frames of reference

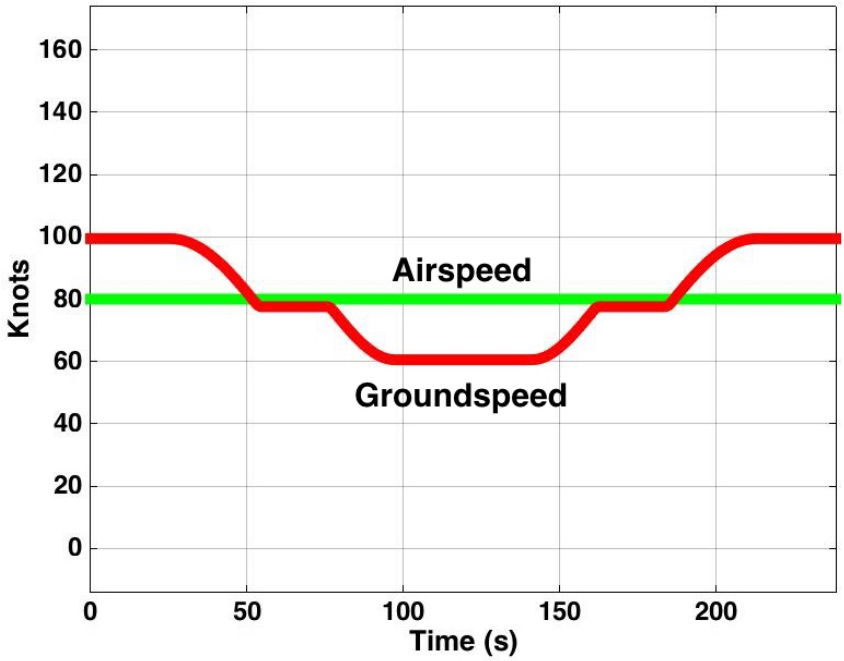


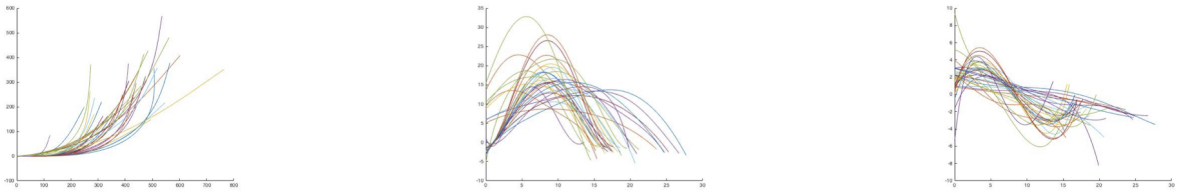
Figure 3.2: Desired airspeed profile in the presence of wind while the helicopter is turning. Note how the airspeed remains constant while the groundspeed changes as the helicopter turns.

3.2 Optimal Speed Profiles

As outlined in the previous section, aerial vehicles such as helicopters must deal with two frames of reference - the airframe and the groundframe. Consequently, both the *airspeed* (in the airframe) and the *groundspeed* (in the ground-frame) are important quantities. The aviation community cares a lot about the airspeed, since most of the vehicle's dynamic limits are defined with respect to the airspeed. For instance, the amount of torque available to a helicopter for climbing directly depends on its airspeed. As a result, it is essential to respect dynamic limits on airspeed and its derivatives. Furthermore, pilots prefer to maintain *smooth* airspeed profiles, where smoothness has two aspects-

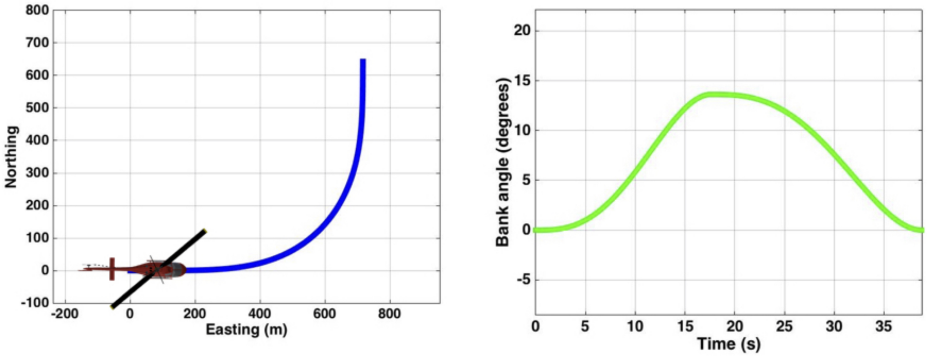
- Respect limits on acceleration and jerk.
- Maintain constant airspeed if possible, even when turning in the presence of wind, as shown in Fig. 3.2.

3.3 Optimal Turn Profiles



Path: y v/s x Bank angles v/s time Roll rate v/s time

(a) Profiles of turns extracted from flight data. The plot on the left shows spatial (Y vs X) profiles, the plot in the middle shows roll vs time, and the plot on the right shows roll-rate vs time.



(b) Spatial and roll profiles for a single 90° turn.

Figure 3.3

Similar to an automobile driving on a freeway, smooth turns are critical to the flight perfor-

mance of helicopters and fixed-wing aircraft. Turns bring into play a lot of the dynamic limits of these systems - airspeed, acceleration, roll and its deriavtives. They often act as bottlenecks for speed (consider slowing a car down before a turn), and are difficult to plan in the presence of wind (consider trying to turn a car at speed while its wheels are slipping on a muddy road). As we will describe later, turning in wind also makes it extremely non-trivial to decouple path from velocity, which is a staple approach to such trajectory optimization problems.

At high speeds ($> 10m/s$), helicopters and fixed-wing aircraft effect a heading change by *rolling* (or *banking*; both terms are used interchangeably in this thesis). The rate of heading change depends on the bank angle, as described in Sec. 5.3. Pilots thus prefer to maintain smooth bank profiles, and limits on bank and bank-rate are safety-critical dynamic limits. Banking a helicopter is akin to steering a car, in that there is a gradual transition to some fixed maximum bank angle (equivalent to the steering angle for cars), holding the maximum bank angle for some time, and a gradual transition back down to zero bank angle. Example turn profiles, extracted from various flight tests and post-processed with a smoothing filter, are shown in Fig. 3.3a. A single pilot-like turn is shown in Fig. 3.3b. Fig. 3.4 shows how a helicopter is able to execute sharped turns when facing into the wind, and must execute shallower turns when facing downwind. This behavior is replicated by our approach.

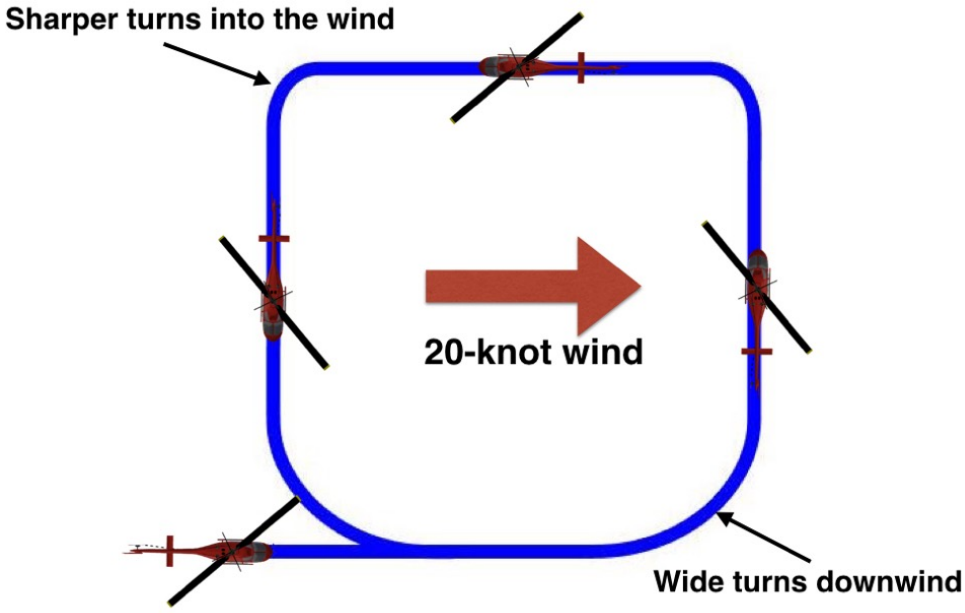
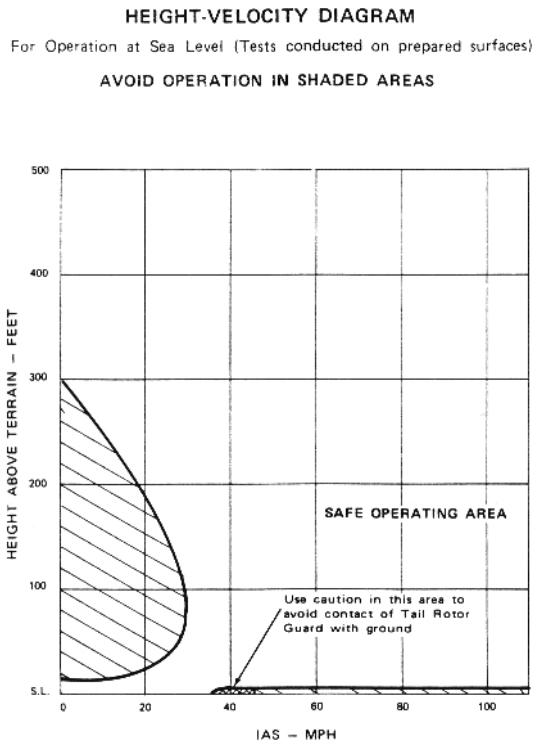


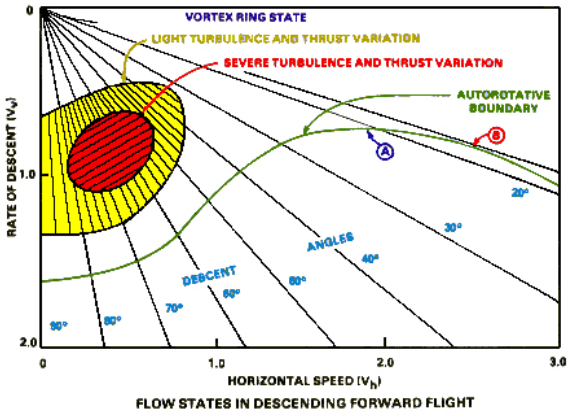
Figure 3.4: Executing turns in wind.

3.4 Takeoff and Landing

While the previous sections described pilot-like behavior for enroute flight, let us now consider the constraints to be kept in mind while taking off and landing. While helicopters can be more agile than fixed-wing aircraft, pilots do prefer somewhat similar takeoff and landing profiles for



(a) H-V curve



(b) Ring vortex state

both systems. Fig. 3.5a shows one of the important system-specific charts referred to by helicopter pilots while taking off and landing. The chart describes a safe height-airspeed region – operation outside this envelope might result in catastrophic failure, and incursions into unsafe height-airspeed regimes must necessarily be minimized. Similarly, Fig. 3.5b shows safe operating regions while landing a helicopter; once again, operation outside this region might cause the helicopter to enter into a potentially hazardous regime known as the *ring-vortex state*. A pilot attempts to respect these charts while taking off and landing.

Chapter 4

Problem Definition

This chapter introduces the problem more concretely, discussing the abstractions that we use. It clearly defines inputs, outputs and constraints, sets up the overall optimization problem and describes the system model that we consider.

We decouple the Z-axis from the X-Y plane, and solve for the Z-profile after solving for the trajectory in X and Y. This allows us to cleanly separate turn-related constraints from climb constraints. Available engine torque limits simultaneous vertical and horizontal accelerations, and appropriate values for the dynamic limits allow us to solve for a Z-velocity profile independent of the X-Y velocity profile. For simplicity of exposition, this section therefore outlines the 2D optimization problem, and a subsequent section (Sec. 5.6) describes how the Z-profile is obtained.

4.1 Model and Dynamics

Aircraft such as helicopters and fixed-wing planes that execute coordinated turns can be described by the fixed-wing UAV model with zero side-slip. In order to describe the dynamics of a fixed-wing UAV in wind, we need to define two coordinate frames - $\mathcal{A}$ and $\mathcal{G}$ as shown in Fig. 3.1. The state space dynamics are defined in $\mathcal{A}$. A state space trajectory in $\mathcal{A}$ can be projected to $\mathcal{G}$ using the wind.

Let the $\mathbb{R}^7$ state space defined in airframe $\mathcal{A}$ be $X = [x, y, v, a, \psi, \phi, \omega]^T$. Let the $\mathbb{R}^2$ controlspace be $U = [j, \alpha]^T$. The dynamical equations are-

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{v} \\ \dot{a} \\ \dot{\psi} \\ \dot{\phi} \\ \dot{\omega} \end{bmatrix} = \begin{bmatrix} v\cos\psi \\ v\sin\psi \\ a \\ j \\ \frac{g\tan\phi}{\sqrt{\dot{x}^2+\dot{y}^2}} \\ \omega \\ \alpha \end{bmatrix} \tag{4.1}$$

We impose a set of bounds on the state and control variables $\{v_{\max}, a_{\max}, j_{\max}, \phi_{\max}, \dot{\phi}_{\max}, \ddot{\phi}_{\max}\}$ (where $|\omega| \leq \dot{\phi}_{\max}, |\alpha| \leq \ddot{\phi}_{\max}$).

Let $\sigma(t) = \{x(t), y(t), \psi(t), \phi(t)\}$ be a time parameterized trajectory defined on the time interval $[0, t_f]$ in the airframe $\mathcal{A}$. The dynamics (4.1) and limits are translated into higher order constraints and bounds on $\sigma(t)$. Without loss of generality, we assume that wind is along the x-axis and has a magnitude v_w. Let $\sigma_g(t) = \{x_g(t), y_g(t), \psi_g(t), \phi(t)\}$ be the trajectory in groundframe g. Let $\sigma_g(t) = \texttt{Proj}(\sigma(t), v_w)$ be a projection function that is defined as follows-

$$\begin{aligned}
\dot{x_g}(t) &= \sqrt{\dot{x}^2(t) + \dot{y}^2(t)} \cos\psi(t) + v_w \\
\dot{y_g}(t) &= \sqrt{\dot{x}^2(t) + \dot{y}^2(t)} \sin\psi(t) \\
x_g(t) &= \int_0^t \dot{x_g}(t)\,\mathrm{dt} \\
y_g(t) &= \int_0^t \dot{y_g}(t)\,\mathrm{dt} \\
\psi_g(t) &= \tan^{-1}\left(\frac{\dot{y_g}(t)}{\dot{x_g}(t)}\right) \\
\phi_g(t) &= \phi(t)
\end{aligned} \tag{4.2}$$

4.2 Inputs and outputs

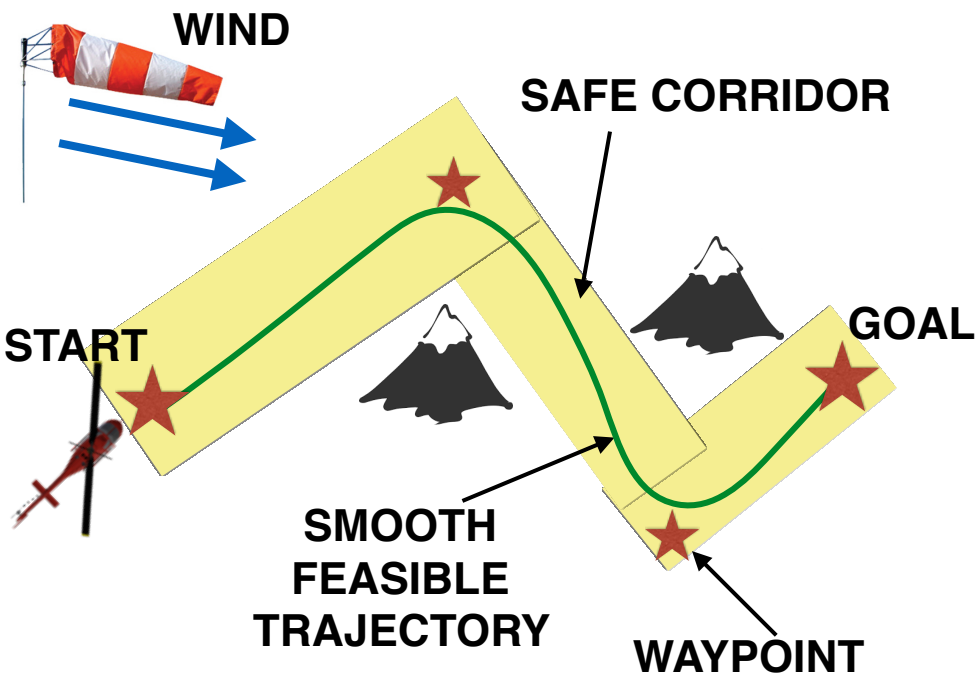


Figure 4.1: The trajectory optimization problem.

Let V_{start} and V_{goal} be the specified start and goal velocities. The input mission, shown in Fig. 4.1, consists of N waypoints $\{w_1, \ldots, w_N\}$, including the start and goal points. These waypoints define $N-1$ segments. A safe flight corridor is specified for each segment. The

function $\mathbb{I}_i(p) \in \{0,1\}$ indicates if the x, y value of a configuration p lies in corridor i. The start and goal positions, together with the corridors, comprise the *spatial constraints*. A maximum segment velocity is specified for each segment $V_{\text{st},i}$. If a configuration belongs to a corridor, it must satisfy the segment velocity limit. The corridors and waypoints are either specified by a human, or are determined by a route-planning module that is invoked prior to calling κITE . The system's dynamic limits, including limits on airspeed, acceleration, jerk, roll, roll-rate, roll-acceleration, and climb-rate, are also specified as inputs. A wind estimation module reports the 3D wind conditions (speed and direction) for any x, y, z point.

The trajectory optimization module outputs a trajectory that satisfies all spatial and dynamic constraints while attempting to be time-efficient (though time optimality is not formally guaranteed).

4.3 Optimization Problem

The complete optimization problem can be formally stated as follows.

$$
\begin{aligned}
\min_{\sigma(.),t_f} \quad & t_f \\
\text{s.t} \quad & \left.\begin{aligned}
\sqrt{\dot{x}^2(t)+\dot{y}^2(t)} &\geq v_{\min} \\
\sqrt{\dot{x}^2(t)+\dot{y}^2(t)} &\leq v_{\max} \\
\sqrt{\ddot{x}^2(t)+\ddot{y}^2(t)} &\leq a_{\max} \\
\sqrt{\dddot{x}^2(t)+\dddot{y}^2(t)} &\leq j_{\max} \\
|\phi(t)| &\leq \phi_{\max} \\
\left|\dot{\phi}(t)\right| &\leq \dot{\phi}_{\max} \\
\left|\ddot{\phi}(t)\right| &\leq \ddot{\phi}_{\max}
\end{aligned}\right\} \text{Derivative Bounds} \\
& \left.\begin{aligned}
\psi(t) &= \tan^{-1}\left(\frac{\dot{y}(t)}{\dot{x}(t)}\right) \\
\dot{\psi}(t) &= \frac{g\tan\phi}{\sqrt{\dot{x}^2(t)+\dot{y}^2(t)}}
\end{aligned}\right\} \text{Dynamics Constraints} \\
& \left.\begin{aligned}
\sqrt{\dot{x}^2(0)+\dot{y}^2(0)} &= V_{start} \\
\sqrt{\dot{x}^2(t_f)+\dot{y}^2(t_f)} &= V_{goal} \\
\sum_{i=1}^{N-1} \mathbb{I}_i\left(\texttt{Proj}\left(\sigma(t), v_w\right)\right) &> 0 \\
\left(\sqrt{\dot{x}^2(t)+\dot{y}^2(t)}\right)\mathbb{I}_i(\sigma(t)) &\leq V_{\text{st},i}, \\
\text{for } i &\in \{1,\ldots,N\}
\end{aligned}\right\} \text{Route Constraints} \\
& \qquad \text{for } t \in [0, t_f]
\end{aligned}
\tag{4.3}
$$

Chapter 5

κITE

This chapter introduces κITE , our trajectory optimization algorithm.

5.1 Motivation

The trajectory optimization problem outlined in 4.3 is non-convex and non-linear because of the presence of roll-rate and roll-acceleration constraints, and because of the disturbance introduced by wind. Any approach to this problem would therefore necessarily reach some local minima. The staple approach to such problems is to first solve for a geometric path respecting spatial constraints, and then solve for a dynamically-feasible velocity profile. However, wind introduces a time-dependent drift, and makes this decoupling extremely non-trivial. It is essential to guarantee that there exists some velocity profile that can be applied to the geometric path without violating dynamic limits. A second challenge is having to simultaneously deal with two frames of reference while satisfying constraints – dynamic limits exist in the air frame (which is moving), while spatial constraints exist in the ground frame. Finally, the optimization procedure must run online in near real-time to account for changing environmental and mission-related conditions.

5.2 Overview

To make the solution-search tractable, we decouple the optimization problem into a path optimization and a velocity profile optimization ([6], [5], [16]). The optimization approach, summarized in Fig. 5.1, proceeds in 4 stages:

1. Phase A: *Path Optimization*. This phase solves for a path that is *guaranteed to be feasible* (in terms of dynamics and route constraints) for a range of constrained velocity profiles. The path is parameterized as a sequence of *sections* which are either straight lines or arcs. The optimizer solves for each section i independently along with a corresponding $V_{\lim,i}$, such that the section is feasible for any velocity profile limited by $V_{\lim,i}$. In the interest of time-optimality, the objective of this optimizer is to maximize the velocity limit $V_{\lim,i}$ while keeping the total arclength of the section small.
2. Phase B: *Velocity Optimization*. This phase optimizes velocity at specific *control points*

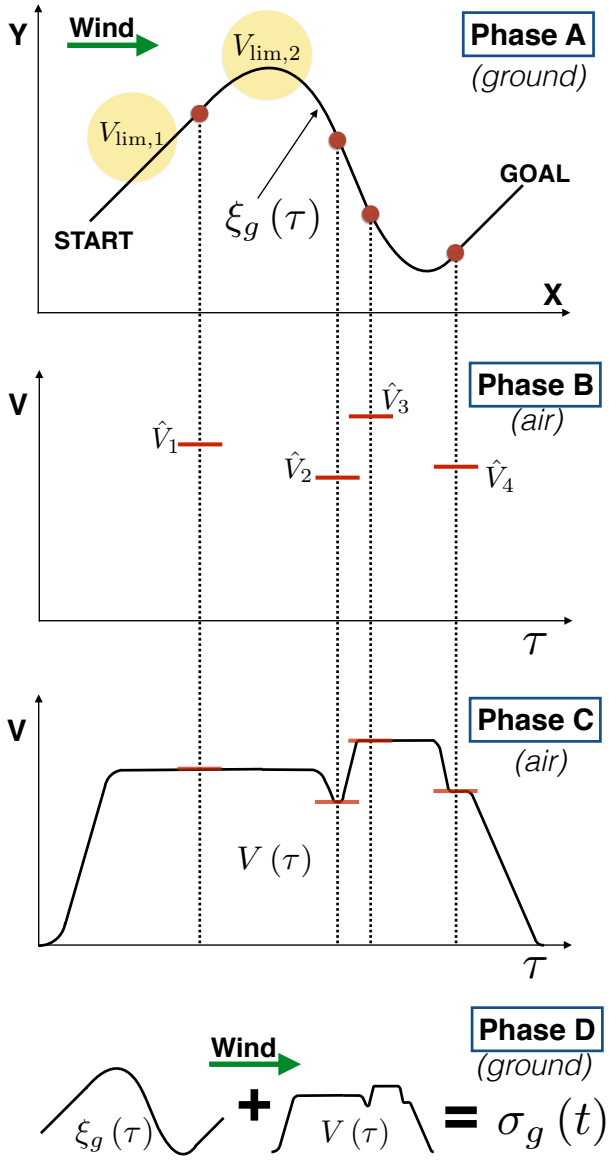


Figure 5.1: Overview of κITE

at the end of the sections to minimize time. By ignoring jerk constraints at this stage and assuming a trapezoidal velocity profile, we are able to solve this optimization very efficiently.

3. Phase C: *Velocity Spline Fitting*. This phase solves for smooth velocity splines that introduce jerk limits.
4. Phase D: *Ground Frame Trajectory Repair*. This phase combines the path from Phase A with the velocity profile from Phase C to yield the final ground frame trajectory.

For simplicity of exposition, we drop the z coordinate from our formulation and note that the Z-profile is solved independent of the X-Y profile while accounting for wind in a similar manner as described in later sections.

5.3 Phase A

Let $\xi(\tau) = \{x(\tau), y(\tau), \psi(\tau)\}$ be a *path* defined over $\tau \in [0, 1]$.

The first stage of the algorithm solves for a ground-frame path $\xi_g(\tau) = \{x_g(\tau), y_g(\tau), \psi_g(\tau)\},\ \tau \in [0, 1]$ that respects corridor constraints. Path optimization is a challenging problem because of a number of reasons - it must guarantee that the path will be dynamically feasible when the velocity profile is determined subsequently and reason about time optimality. Moreover, the presence of wind as a forcing function breaks the necessary decoupling between path and time, and must be dealt with in a principled manner. Our solution structure addresses these concerns.

Parameterization

Solving for arbitrary path shapes is intractable, especially with lots of waypoints and large segment lengths. We restrict our solution to the space of ground-frame straight lines ξ_{st} and arcs ξ_{arc} with smooth curvature profiles, where the path is a sequence $\{\xi^1_{\text{st}}, \xi^1_{\text{arc}}, \xi^2_{\text{st}}, \xi^2_{\text{arc}} \ldots\}$. Arc end-points of ξ^i_{arc} lie on the lines defined by (w_i, w_{i+1}) and (w_{i+1}, w_{i+2}) respectively. This parameterization also scales well with problem size. Instead of searching for individual points along the path, only the arcs need to be explicitly determined, and the straight segments simply connect their endpoints.

Arc Optimization

Arcs are determined for every ordered triple of waypoints $(w_{i+1}, w_{i+2}, w_{i+3})$. Limits on ϕ (roll), $\dot{\phi}$ and $\ddot{\phi}$ become active along the arc, and must be satisfied. These constraints directly limit the arc's curvature κ and its derivatives with respect to arclength (κ', κ''). It is essential to note that these curvature limits are *not invariant* - it can be shown that they also depend on airspeed and acceleration according to the function `CurvLimit` $\left(v_{\max}, \phi_{\max}, \dot{\phi}_{\max}, \ddot{\phi}_{\max}, a_{\max}\right)$ (proof in

Appendix A):

$$\kappa_{max} = \frac{g \tan \phi_{\max}}{v_{\max}^2}, \quad \kappa'_{max} = \frac{\alpha}{\gamma^3} - \frac{\beta}{\gamma^2}, \quad \kappa''_{max} = \frac{g\ddot{\phi}_{\max}}{v_{\max}^4}$$
$$\left(\alpha = g\dot{\phi}_{\max}, \quad \beta = 2\kappa_{max} a_{\max}, \quad \gamma = \frac{3\alpha}{2\beta}\right) \tag{5.1}$$

Solving for a curvature profile that respects these limits guarantees that the path will be dynamically feasible when the velocity profile is subsequently optimized with the same limits on airspeed and acceleration, which allows us to effectively *decouple solving for the path and velocity in a principled way*. Since these dynamic limits are defined in the air-frame, our approach solves for these curvature profiles in the airframe and projects them into the ground-frame.

Since curvature is constrained by airspeed, arcs function as velocity bottlenecks. To enforce time optimality, we determine the maximum airspeed at which we can perform a turn with the least curvature, while respecting corridor constraints.

Finally, arcs are meant to carry out heading changes in the ground frame while operating in the air frame. As an analogue, consider rowing a boat across a river to a point on the opposite bank - the boat must necessarily be oriented such that some component of its velocity zeros out the flow of the river. Similarly, to achieve a desired groundframe heading ψ, aerial vehicles must be oriented at an angle ψ_g to counteract the effect of wind. To convert between ψ_g and ψ, we use the function `HeadingInAir` (ψ_g, v, v_w):

$$\psi = \operatorname{acos}\left(-\frac{v_w \sin(\psi_g)}{v}\right) - \frac{\pi}{2} + \psi_g \tag{5.2}$$

Algorithm 1 captures this process, while the following subsections explain how we determine arcs that effect a required heading change.

Arc Parameterization

Let s be the arc-length of a path. Let $\xi(s)$ be an arclength parameterized path. Building on [20], each arc ξ^i_{arc} is represented as a C2 curvature spline $\kappa(s)$ comprising a degree-4 polynomial $\kappa_1(s)$, a constant-curvature section $\kappa_2(s) = \kappa_{trans}$ and another degree-4 polynomial $\kappa_3(s)$. S_f^1, S_f^2, S_f^3 denote the arclengths of the three sections, and S_f denotes the total arclength. Representing arcs using three curvature segments allows us to replicate the pilot-like behavior of transitioning into a turn, holding a steady bank angle, and transitioning out of a turn. These curvature primitives can be spliced together to result in more complex maneuvers. Fig. 5.2 shows an example of these curvature-parameterized turns.

At this stage of the algorithm, we assume a constant velocity $V_{\text{arc},i}$ along the arc to determine its ground-frame shape. This velocity serves as the upper limit for this segment in a subsequent velocity optimization. We can recover a ground-frame path $\xi_g(s)$ from $\kappa(s)$ using the function

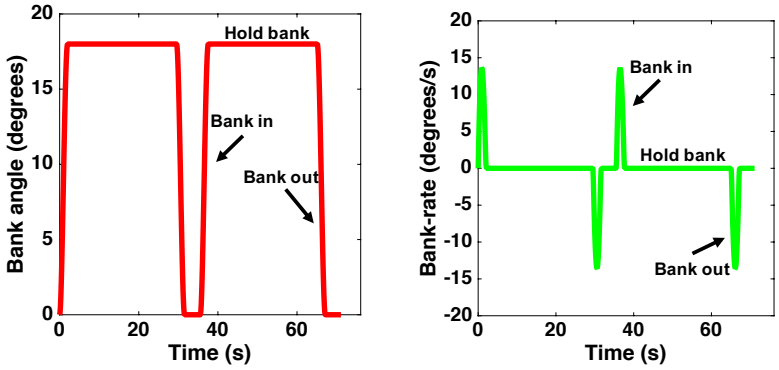


Figure 5.2: Bank and bank-rate profiles for two turns parameterized using curvature splines. Note the transitions in and out of maximum bank.

`CurvPolyGnd` $(\kappa(s))$:

$$\begin{aligned}
\psi(s) &= \int_0^s \kappa(s) \\
x(s) &= \int_0^s cos(\psi(s)) + \frac{v_w}{V_{\text{lim},i}} \int_0^s ds \\
y(s) &= \int_0^s sin(\psi(s)) \\
\psi_g(s) &= \tan^{-1}\left(\frac{\dot{y}(s)}{\dot{x}(s)}\right)
\end{aligned} \tag{5.3}$$

It is now trivial to carry out a change of index from $s \in [0, S_f]$ to $\tau \in [0, 1]$ by setting $\tau(s) = s/S_f$ and obtain $\xi_g(\tau)$.

To solve for one of the segments of $\kappa(s)$, we use `CurvPoly` $(\kappa_0, \kappa_f, S_f, \kappa_{max}, \kappa'_{max}, \kappa''_{max})$:

$$\begin{aligned}
\text{find} \quad & \kappa(s) \\
& \kappa(0) = \kappa_0, \kappa(S_f) = \kappa_f \\
& \kappa'(0) = 0, \kappa'(S_f) = 0 \\
& \kappa(s) \leq \kappa_{max}, \kappa'(s) \leq \kappa'_{max}, \kappa''(s) \leq \kappa''_{max}
\end{aligned} \tag{5.4}$$

Algorithm 1, lines 4 to 15 highlight how the spline is constructed to satisfy the required $\Delta\psi$.

We solve this problem as a Quadratic Program.

Final path and velocity limits

Once we have obtained all the arcs, we concatenate straight segments and arcs to yield the final ground-frame path $\xi_g(\tau)$. Each segment also has an airspeed bound defined by the corresponding waypoint definition for ξ^i_{st}, and the airspeed limit imposed by Phase A for ξ^i_{arc}. We thus have a set of airspeed limits $V_{\text{lim}} = \begin{bmatrix} V_{\text{st},1} & V_{\text{arc},1} & \dots & V_{\text{arc},N-2} & V_{\text{st},N-1} \end{bmatrix}$, that are used by Phase B.

Algorithm 1: ArcOpt $(w_i, w_{i+1}, w_{i+2}, v_w)$

for $v \leftarrow [v_{\max}, v_{\min}]$ **do**
  $(\kappa_{max}, \kappa'_{max}, \kappa''_{max}) \leftarrow \texttt{CurvLimit}\left(v, \phi_{\max}, \dot{\phi}_{\max}, \ddot{\phi}_{\max}, a_{\max}\right)$
  $\Delta\psi \leftarrow \texttt{HeadingInAir}\left(\angle(w_{i+1}, w_{i+2})\right) - \texttt{HeadingInAir}\left(\angle(w_i, w_{i+1})\right)$
  for $\kappa \leftarrow [\kappa_{min}, \kappa_{max}]$ **do**
    $\kappa_2(s) = \kappa$
    for $S_f^1 \leftarrow [S_f^{\min}, S_f^{\max}]$ **do**
      $\kappa_1(s) \leftarrow \texttt{CurvPoly}\left(0, \kappa, S_f^1, \kappa_{max}, \kappa'_{max}, \kappa''_{max}\right)$
      if $\kappa_1(s) \in \emptyset$ **then**
        break
    $S_f^3 \leftarrow S_f^1$
    $\kappa_3(s) \leftarrow \texttt{CurvPoly}\left(\kappa, 0, S_f^3, \kappa_{max}, \kappa'_{max}, \kappa''_{max}\right)$
    $S_f^2 = \frac{\Delta\psi - \int_0^{S_f^1} \kappa_1(s) - \int_0^{S_f^3} \kappa_3(s)}{\kappa}$
    if $S_f^2 \geq 0$ **then**
      break
  $\kappa(s) \leftarrow \begin{bmatrix}\kappa_1(s) & \kappa_2(s) & \kappa_3(s)\end{bmatrix}$
  $\xi^i_{\text{arc,gnd}}(s) \leftarrow \texttt{CurvPolyGnd}\left(\kappa(s)\right)$
  if $\sum_{j=i}^{i+1} \mathbb{I}_j\left(\xi^i_{\text{arc,gnd}}(s)\right) > 0$ **then**
    break
return $v, \xi^i_{\text{arc}}(s), \xi^i_{\text{arc,gnd}}(s)$

5.4 Phase B

Time Optimization Problem

This phase determines an optimal scheduling of speeds along a finite set of control points belonging to $\xi_g(\tau)$. The $2N-4$ control points are the start and end points of each turn segment, which divide $\xi_g(\tau)$ into a sequence of straight segments ξ^i_{st} and turns ξ^i_{arc}. We obtain the segment velocity limits V_{lim} from Phase A, along with the air-frame path lengths S_i for each segment. We further assume an acceleration $\hat{a}_{\max} = a_{\max} - \varepsilon_{\text{tol}}$ which is lower than the acceleration limit of the system. While the current phase ignores jerk, using a lower acceleration at this stage allows us to fit a jerk-limited velocity spline at a later stage. The optimization problem now is to determine the control-point velocities $\{\hat{V}_i\}$ which minimize time (where $\hat{V}_0 = V_{start}, \hat{V}_{N+1} = V_{goal}$):

$$\begin{aligned} \underset{\{\hat{V}_i\}}{\text{minimize}} \quad & t_f\left(\{\hat{V}_i\}, \{S_i\}, \hat{a}_{\max}\right) \\ \text{subject to} \quad & \hat{V}_i \leq V_{\text{lim},i} \\ & \frac{|\hat{V}_{i+1}^2 - \hat{V}_i^2|}{2\hat{a}_{\max}} \leq S_i \end{aligned} \tag{5.5}$$

Algorithm 2: VelOpt($V_{goal}, V_{start}, \{V_{\lim,i}\}, \{S_i\}, a$)

$\left\{\hat{V}_i\right\} \leftarrow \{V_{start}, \{V_{pt,i}\}, V_{goal}\}$; $Visited \leftarrow \{0\}_{N+2}$;
$\hat{V}_1 \leftarrow$ `MakeFeasible` $\left(\hat{V}_0, \hat{V}_1, a, S_1\right)$; $\hat{V}_N \leftarrow$ `MakeFeasible` $\left(\hat{V}_{N+1}, \hat{V}_N, a, S_N\right)$;
$Visited[0] \leftarrow 1$; $Visited[N+1] \leftarrow 1$;
repeat
  $i \leftarrow Minimum\left(\left\{\hat{V}_i\right\}\right)$ $s.t.$ $Visited[i] = 0$;
  if $Visited[i-1] = 0$ **then**
    $\hat{V}_{i-1} \leftarrow$ `MakeFeasible` $\left(\hat{V}_i, \hat{V}_{i-1}, a, S_i\right)$
  if $Visited[i+1] = 0$ **then**
    $\hat{V}_{i+1} \leftarrow$ `MakeFeasible` $\left(\hat{V}_i, \hat{V}_{i+1}, a, S_{i+1}\right)$
  $Visited[i] \leftarrow 1$;
until $Visited[0..N+1] = 1$;

The total time t_f is defined as follows-

$$t_f = \sum_i t_i \tag{5.6}$$

Given a pair of consecutive point velocities $\hat{V}_i, \hat{V}_{i+1}$, t_i can be computed according to the following-

$$V_{mid} = \min\left(\sqrt{\frac{2\hat{a}_{\max}S_i + \hat{V}_i^2 + \hat{V}_{i+1}^2}{2}}, V_{\lim,i}\right) \tag{5.7}$$

$$t_i = \frac{2V_{mid} - \hat{V}_{i+1} - \hat{V}_i}{\hat{a}_{\max}} + \tag{5.8}$$

$$\left(S_i - \frac{2V_{mid}^2 - \hat{V}_{i+1}^2 - \hat{V}_i^2}{2\hat{a}_{\max}}\right)\frac{1}{V_{mid}} \tag{5.9}$$

Algorithm for Initialization

Algorithm 2 is used to feasibly initialize the nonlinear optimization problem above, which results in significant improvements in convergence rates. It uses the function `MakeFeasible` $\left(\hat{V}_1, \hat{V}_2, a, S\right)$, defined as:

$$\hat{V}_2 = \sqrt{\hat{V}_1^2 + 2aS} \tag{5.10}$$

5.5 Phase C

This stage operates on $\{\hat{V}_i\}$ and fits a smooth, jerk-limited spline $V_i(t)$ between each $\hat{V}_i$ and $\hat{V}_{i+1}$. $V_i(t)$ is derived by integrating a C1 acceleration spline $a(t)$ comprising a degree-3 polynomial

$a_1(t)$, a constant-acceleration section $a_2(t) = a_{trans}$ and another degree-3 polynomial $a_3(t)$. t_f^1, t_f^2, t_f^3 denote the time spanned by the three sections, and t_f denotes the total time of the spline. Each acceleration spline segment effects a velocity change from some $\hat{V}_i$ to $\hat{V}_{i+1}$, and is exactly analogous in structure to the curvature spline described earlier. This allows us to simulate a pilot-like smooth acceleration ramp-up, holding a steady acceleration and a smooth ramp-down to achieve the desired velocity change. We omit the details of computing these splines, since they are the same as for the curvature splines.

Once all the spline segments have been computed, they are combined to yield the final airspeed spline $V(t), \quad t \in [0, t_f]$. It is important to note that time has been used here merely as a suitable parameter to compute these splines, and that it *does not* represent the actual time profile of the trajectory. $V(t)$ is thus converted to $V(\tau)$ by setting $\tau = \frac{t}{t_f}$, and consistency of τ with Phase A is maintained by construction. The next stage computes the final time-parameterized trajectory.

5.6 Phase D

At this stage, we have a ground-frame path $\xi_g(\tau)$ (Phase A), a ground-referenced heading profile $\psi_g(\tau)$ (Phase A) and an airspeed profile $V(\tau)$ (Phase C). We obtain a ground-referenced, time-parameterized trajectory $\sigma_g(t)$ according to the following-

1. Obtain a groundspeed profile:

$$v_g(\tau) = \sqrt{V(\tau)^2 - v_w^2 \sin^2(\psi_g(\tau))} + v_w \cos(\psi_g(\tau)) \tag{5.11}$$

2. Obtain the *ground-frame* distance profile:

$$S_g(\tau) = S_g(x(\tau)) = \int_0^{x(\tau)} \sqrt{1 + y'(x)^2} dx, \quad y' = \frac{dy}{dx} \tag{5.12}$$

3. Compute the time profile:

$$t(\tau) = \int_0^{S_g(\tau)} \frac{dS_g}{v_g(\tau)} \tag{5.13}$$

   At this point, it is trivial to replace τ with $t(\tau)$, and obtain a time-parameterization.

4. Compute the *air-referenced* yaw profile:

$$\psi(t) = \operatorname{atan}\left(\frac{v_g(t)\sin(\psi_g(t))}{v_g(t)\cos(\psi_g(t)) - v_w}\right) \tag{5.14}$$

5. Compute the roll profile:

$$\phi_g(t) = \phi(t) = \operatorname{atan}\left(\frac{V(t)\dot{\psi}(t)}{g}\right) \tag{5.15}$$

Each waypoint also has an associated height that must be achieved. We compute the Z-profile $z_g(t)$ using the timing information from the XY trajectory. This timing information allows to determine the time available for carrying out height changes between waypoints. The Z-profile is parameterized as a C2 spline divided into segments that start and end on waypoints. These segments satisfy boundary value constraints, and also respect speed and acceleration limits.

The final ground-frame trajectory $\sigma_g(t) = \{x_g(t), y_g(t), z_g(t), \psi_g(t), \phi_g(t)\}$ is now complete.

5.7 Takeoff

While κITE deals with enroute flight, takeoff and landing are special cases that are dealt with separately.

Takeoff profiles are determined by the height-velocity (H-V) curve for the given helicopter, an example of which is shown in Fig. 3.5a. This curve denotes a safe airspeed-height regime, and operating outside this regime might lead to a potential crash. Pilots therefore attempt to minimize time spent in the unsafe region of the H-V curve. Given wind conditions and an obstacle-checking interface, we iteratively compute takeoff profiles of increasing aggressiveness until a collision-free takeoff is found. Takeoff profiles are computed using a forward simulation in two phases-

1. The *translational lift* phase, where the system builds up forward airspeed without climbing beyond the limits of the H-V curve. This is enforced by limiting the vertical acceleration available to the system.
2. The *ascend* phase, where the system has access to greater vertical acceleration. This acceleration limit is determined by a dynamical model that reasons about available engine power given operating conditions.

While the nominal takeoff profile avoids the unsafe H-V region completely, more aggressive profiles briefly encroach into unsafe H-V regions to facilitate takeoff in more confined areas. Fig. 5.3 shows examples of such takeoff profiles. Crucially, the algorithm fails if it cannot determine a safe and feasible takeoff profile given the wind conditions and obstacle locations. This allows the autonomous system to abort unsafe missions without even taking off.

5.8 Landing

The H-V curve is an important constraint while landing, as is the onset of a dangerous condition known as a vortex ring state. While landing, pilots try and maintain a nominal *glide slope* of $6 - 12^\circ$, where the glide slope is defined as the ratio of vertical speed to horizontal groundspeed. The trajectory optimization procedure yields the groundspeed profile over the entire trajectory, from the end of takeoff to the point of landing. The vertical velocity profile for the landing segment is then computed according to the glide slope.

To ensure safety, κITE fails if the landing segment is oriented such that the wind is a tailwind. Aerial vehicles typically try and land into the wind, and tailwinds can lead to unstable and

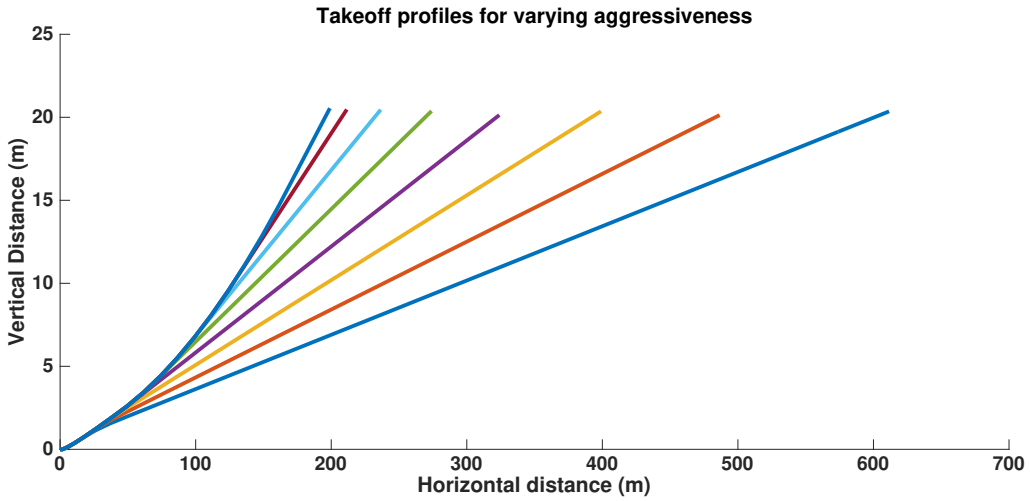


Figure 5.3: Takeoff profiles with varying aggressiveness. Note that the axes are not at the same scale.

potentially dangerous flight. Some example landing profiles with varying glide slopes are shown in Fig. 5.4.

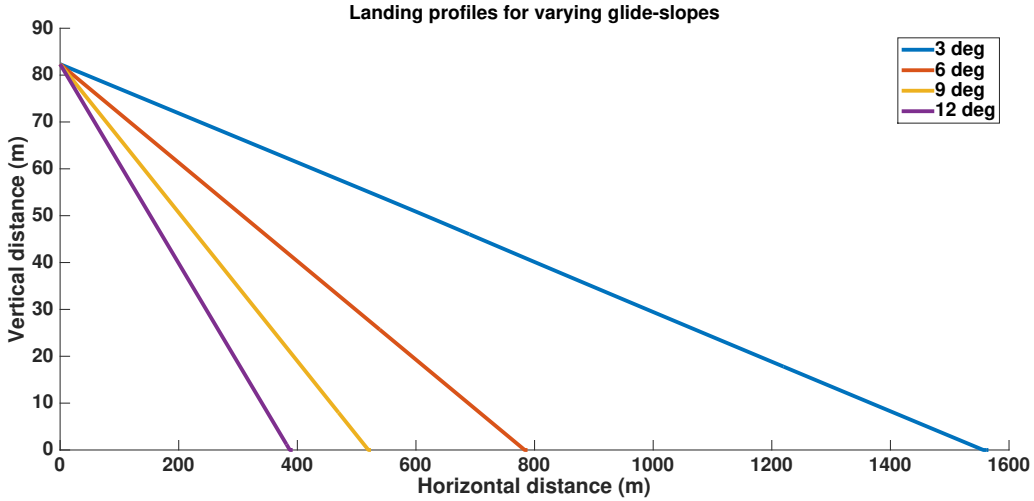


Figure 5.4: Landing profiles with varying glide slopes. Note that the axes are not at the same scale.

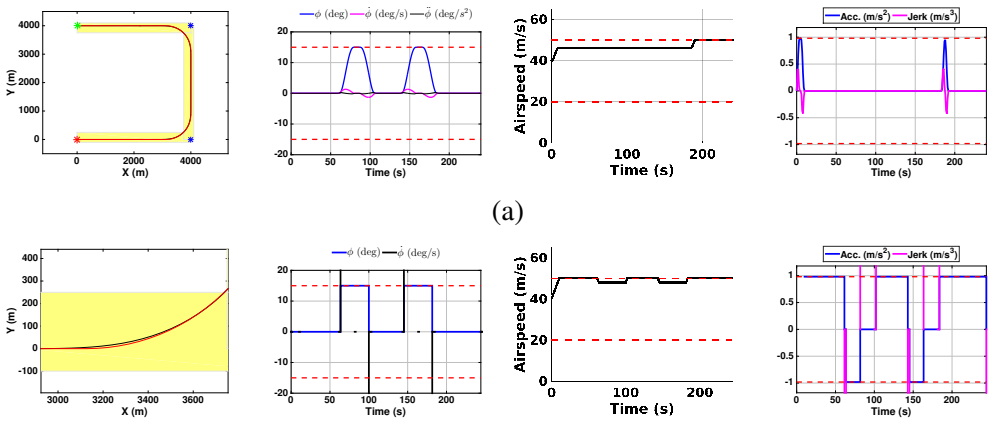
Chapter 6

Results

We will now explore both simulation and experimental results that showcase the efficiency and versatility of our trajectory optimization approach. We have open-sourced a MATLAB implementation of κITE at `https://bitbucket.org/castacks/kite_optimizer`. We compare κITE against a baseline that uses constant-curvature arcs to turn, while trying to maintain as high a speed as possible without violating the roll limit.

6.1 Simulation Results

6.1.1 Solution Quality

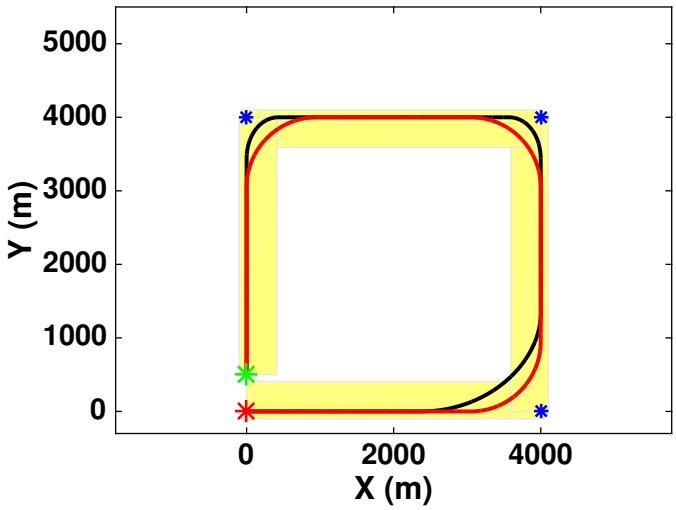


(a)

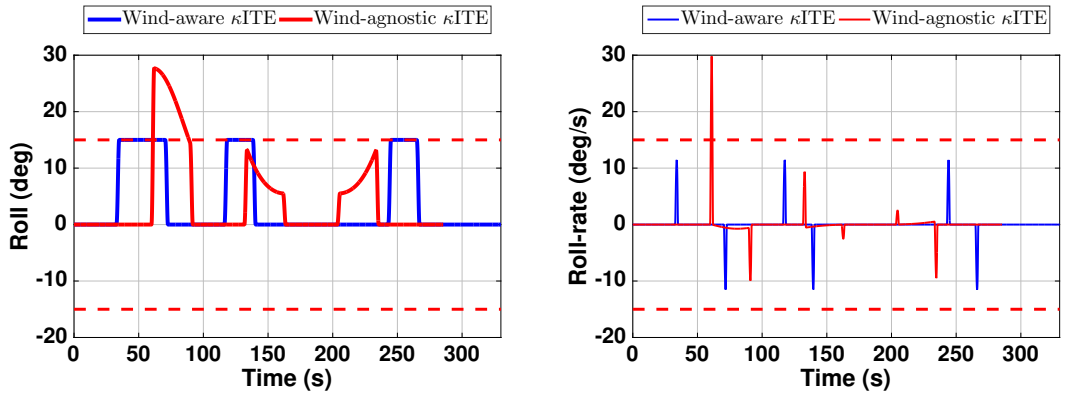
(b)

Figure 6.1: Comparing the spatial, roll, speed and acceleration profiles of κITE (top) with the baseline (bottom). Corridors are highlighted in yellow, and limits are represented by red lines. A blowup of (bottom left) the spatial profiles (top left) shows κITE in black and the baseline in red.

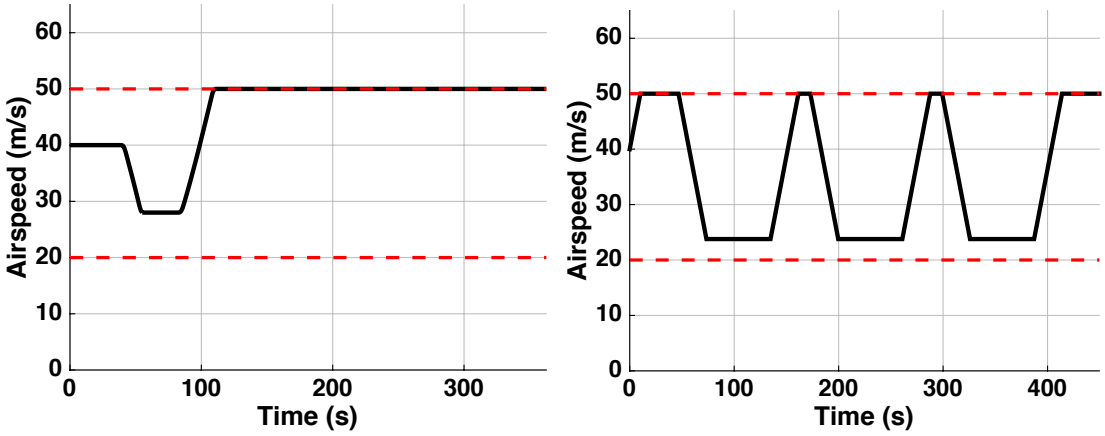
1. **Respecting dynamic limits**
   Fig. 6.1 shows how κITE is able to respect limits on speed, acceleration, jerk, roll, roll-rate and roll-racceleration. This is essential for stable trajectory tracking, especially in the



(a)



(b)



(c)

Figure 6.2: Demonstrating the importance of wind-cognizance in the trajectory planning stage. (a) compares the spatial profiles of wind-aware κITE (black) with a wind-agnostic variant of κITE (red) in the presence of a $20m/s$ wind along the x-axis. A feedback controller used to follow both trajectories in this wind violates roll and roll-rate limits (b) with the wind-agnostic trajectory. (c) shows how the naive baseline (right) has to slow down to execute feasible turns in this wind regime, while κITE (left) is still able to maintain high speeds.

presence of disturbing forces such as wind that might cause the system to exceed its control margin and enter into a potentially hazardous state. Fig. 6.1 compares the output of κITE with the baseline.

2. **Performance With Wind**
   Fig. 6.2 shows a scenario where a helicopter is flying in the presence of a $20m/s$ wind directed along the $+ve$ X-axis. As Fig. 6.2(b) shows, a feedback controller that attempts to follow the trajectory computed without taking wind into account (red trajectory in Fig. 6.2(a)) would have to exceed roll and roll-rate limits at the same airspeed. Wind-cognizant κITE , on the other hand, generates a trajectory that is dynamically feasible in this wind regime. One can see how κITE makes use of wind by generating sharper-looking turns into the wind direction. Similarly, Fig. 6.2(c) shows that the naive baseline must slow down considerably to execute dynamically feasible turns in this wind regime, while κITE is able to maintain high speeds.

6.1.2 Scalability and versatility

κITE can handle very long routes with a variety of segment length ratios. The solution structure is trivially able to accommodate long, straight segments, while algorithms without this structure would struggle to compute such routes in a reasonable amount of time. Fig. 6.3 shows one such situation, in which κITE computes a ~ 290 km with a mixture of long and short segments.

We also tested a $C++$ implementation of κITE on 100 randomly generated problems across three runs with 10, 25 and 50 waypoints respectively. Individual segment lengths range from $300m$ to $5000m$, and are randomly chosen for each waypoint, as are the angles between segments. A pre-computed lookup table is used for quickly determining both curvature and acceleration spline primitives, which allows vastly improved execution times. We report the average execution times for different phases of κITE for both runs (Table 6.1) :

Table 6.1: Execution times (in ms) of all the stages of κITE

Num Waypoints	Phase A	Phase B	Phase C	Phase D
10	203.00	93.26	0.12	1095.84
25	496.10	1054.00	0.23	1067.83
50	1241.41	4785.82	0.48	7121.01

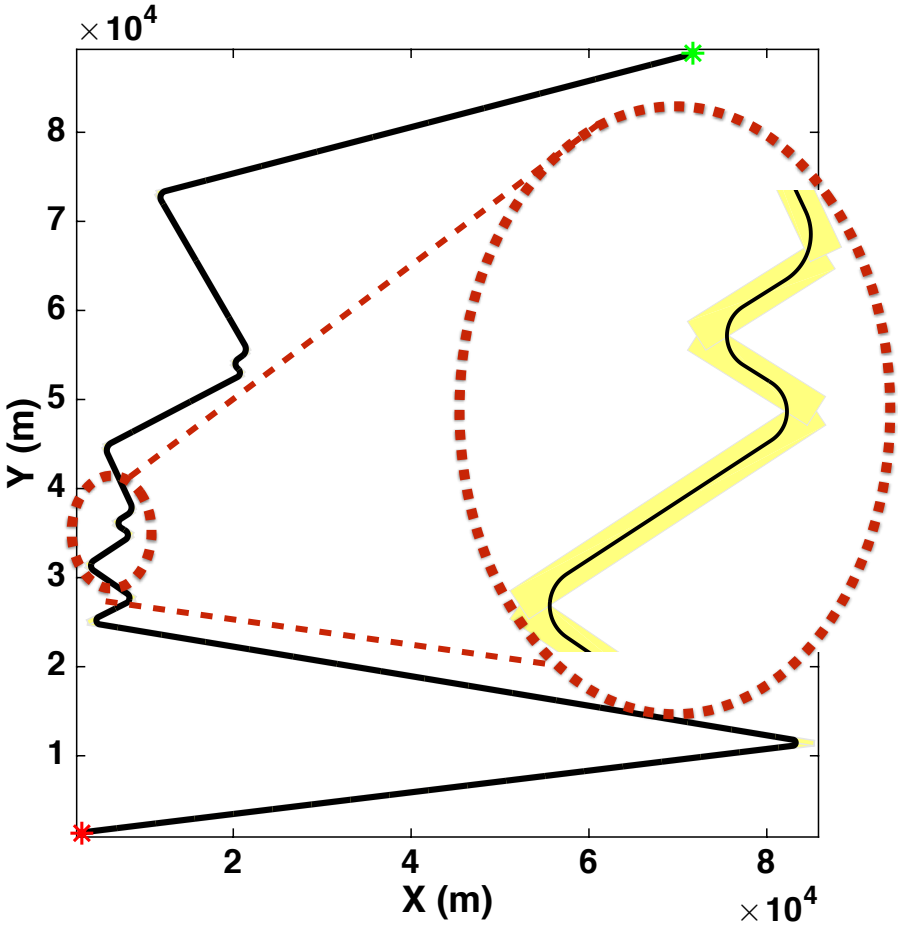


Figure 6.3: A $\sim$ 290 km trajectory computed by κITE . The problem has both long route segments and short segments with closely spaced turns, as the cutout shows. κITE 's solution structure allows it to efficiently deal with such problems. Again, the corridor is highlighted in yellow.

Figure 6.4: Our autonomous helicopter test platform. The laser sensor is visible at the nose of the vehicle.

6.2 Experimental Results

6.2.1 Setup

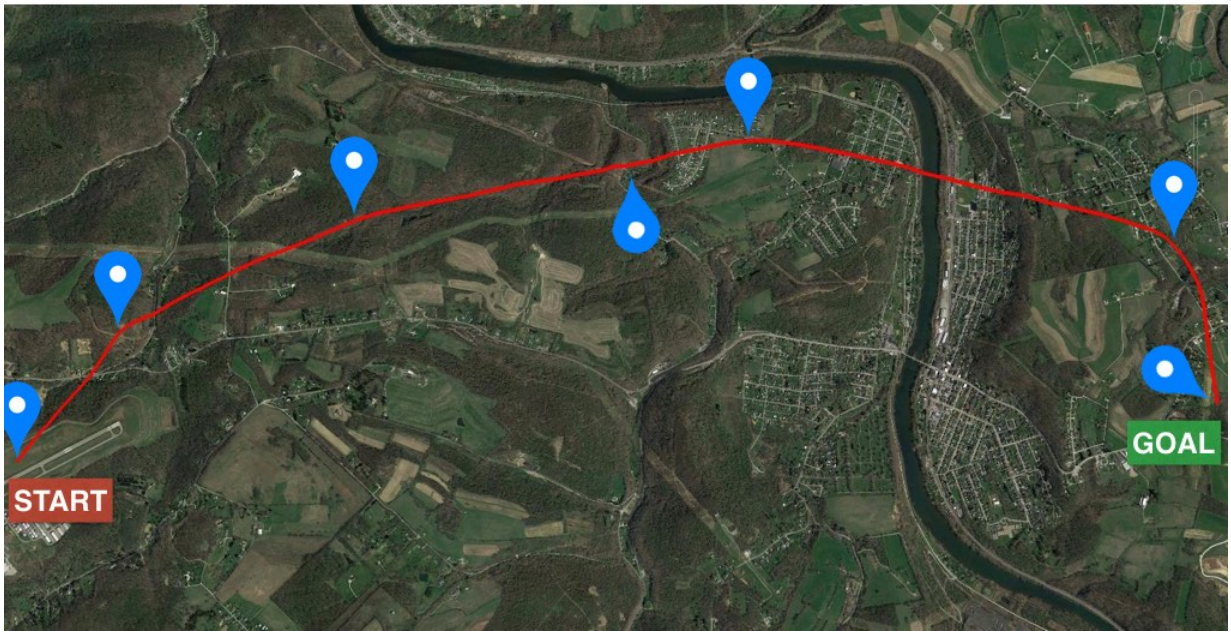
Our algorithm is currently part of the C++-based motion planning architecture running on board a full-scale autonomous helicopter with a human pilot-in-the-loop. The heliopter, shown in Fig. 6.4, is equipped with a scanning laser for perception, along wth inertial sensors and GPS. Its 6D pose and velocity are estimated with the help of both GPS and inertial sensors. κITE serves as a global planner, and computes the entire trajectory from takeoff to landing in real-time whenever the wind conditions or mission requirements change. A local planner [7] is responsible for following this trajectory in the nominal case, and for performing obstacle avoidance should the need arise. The algorithm receives real-time information about wind from an on-board pitot tube, and re-plans whenever the wind changes significantly.

6.2.2 Performance In Wind

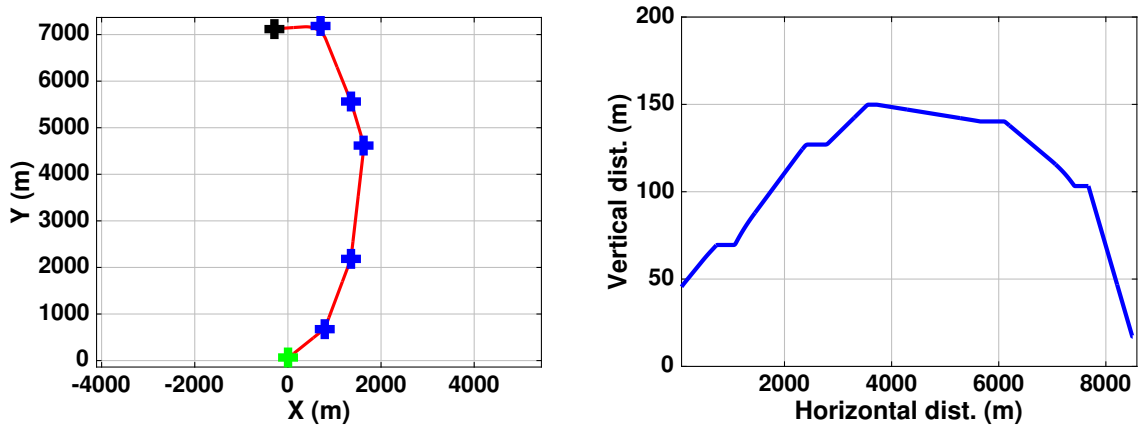
Fig. 6.5 shows the mission waypoints and planned trajectory for a flight test performed with our system in a ~ 20 knots wind blowing towards $270°$N. Fig. 6.6 shows the speed (a), heading (b) and roll (c) profiles for the same mission. The groundspeed and heading were tracked fairly well, while ensuring that the commanded airspeed respected acceleration and jerk constraints of $0.1g$. The commanded roll and roll-rates are within the system limits indicated by the dashed red line. The plots also indicate the direction in which the helicopter must be pointed (i.e the *crab* angle) such that the desired ground-frame heading can still be maintained in the given wind regime. It is to be noted that with zero wind, the airspeed would equal groundspeed, as would crab angle and desired heading. Fig. 6.7 shows similar speed and heading performance plots for another flight test with a ~ 19 knots wind blowing towards $80°$N.

Our system has thus far been tested in 23 flights under winds upto 40 knots, where the algorithm failed to compute a feasible trajectory in 3 cases. The failures were due to very tight corridors, where the algorithm could not keep the trajectory in safe airspace under the corresponding wind conditions. κITE accepts the waypoints and corridors as inputs, and does not

compute them.



(a)

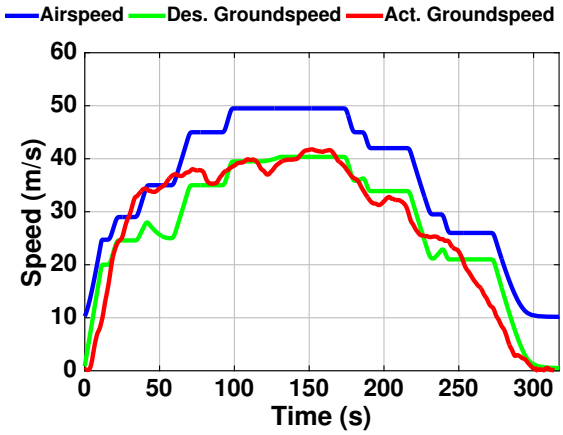


(b)

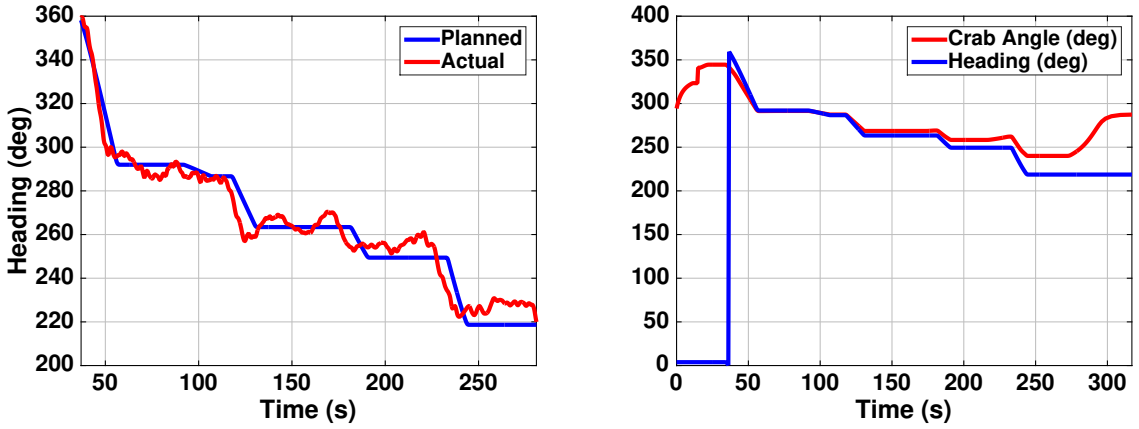
Figure 6.5: Results from a test conducted on our full-size helicopter with a $\sim$ 20 knots wind blowing towards 270°N. (a) shows the complete trajectory overlaid on a map along with the mission waypoints; (b) shows the XY spatial profile of the trajectory (left), along with the vertical profile (right). Waypoints are shown in blue, the start point in green and the final point in black. This is a complete trajectory from takeoff to landing. The safe flight corridor has not been shown due to scale.

6.2.3 Online Re-planning

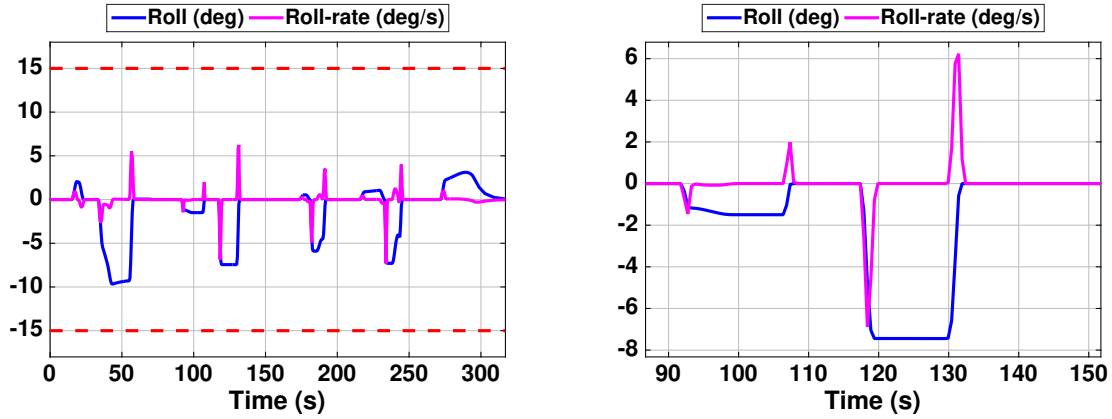
Give changing wind conditions or mission requirements, it is essential for the motion planning architecture to respond and re-plan in near real-time. Fig. 6.8a shows plots of wind direction and magnitude from one of our test flights, and compares the spatial profile of the trajectory (b)



(a)

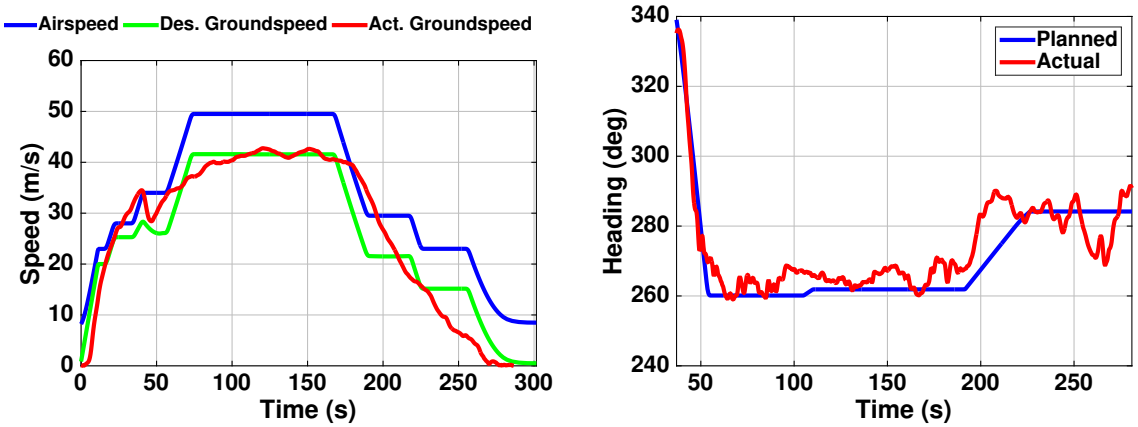


(b)



(c)

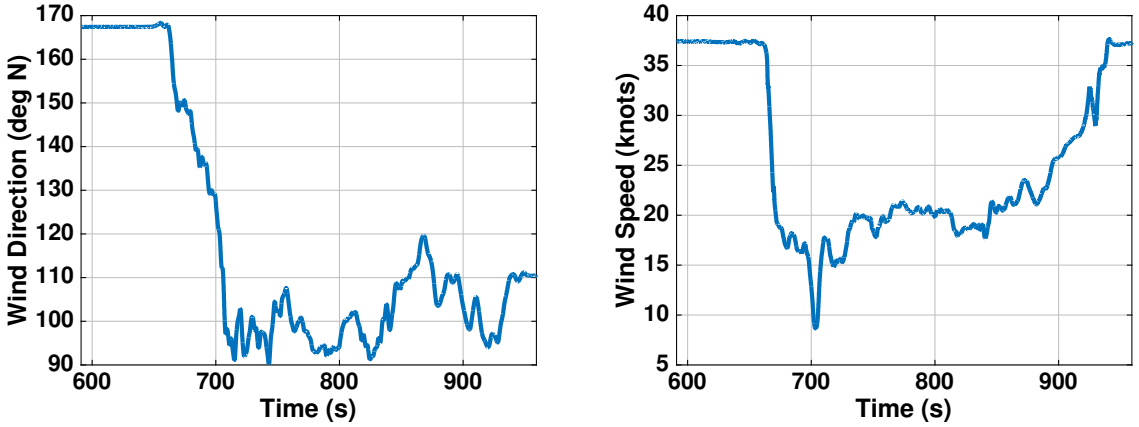
Figure 6.6: Speed, heading and roll profiles for the flight test shown in Fig. 6.5. (a) shows the commanded airspeed, commanded groundspeed and the measured groundspeed. Note that in the absence of wind, airspeed and groundspeed would be equal; (b) shows the commanded and executed heading profile (left), along with the crab angle necessary for maintaining heading in the given wind environment (right); (c) shows the commanded roll and roll-rate profile for the entire trajectory (left), with a magnified view from a section of the trajectory (right). The roll limits are represented by the dashed red line.



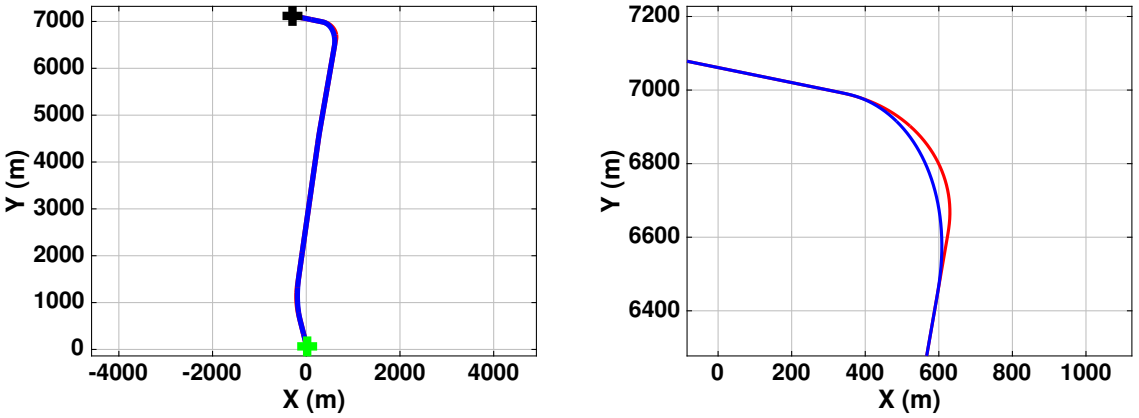
(a)

Figure 6.7: Results from another test flight in the presence of a a $\sim$ 19 knots wind blowing towards $80°$N. Once again, we compare commanded airspeed, commanded groundspeed and measured groundspeed (left), and commanded and measured heading (right).

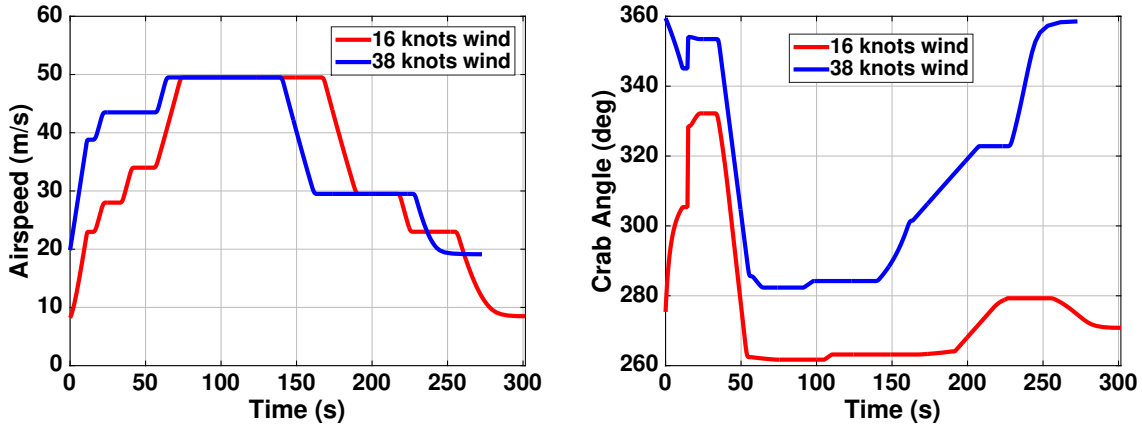
and commanded airspeed and crab angle (c) for two different wind regimes encountered during the test - 38 knots, $170°$N and 16 knots, $90°$N. Such situations are commonly encountered when the helicopter takes off, where the wind conditions at ground level can be quite different from wind conditions at cruising altitude. While κITE accepts estimates of wind (say, from a weather station) all along the mission to compute a feasible trajectory, it is essential for it to re-compute its trajectory online when the measured wind deviates significantly from the original estimates. In all our tests, κITE returns a solution (or failure code in the three runs mentioned above) within $5 - 7s$.



(a)



(b)



(c)

Figure 6.8: Results from a flight test showing online re-planning when the measured wind changes from 38 knots, 170°N to 16 knots, 90°N (a) shows a plot of wind speed and direction estimated in real-time with an on-board pitot tube; (b) shows the full trajectory from start (green) to goal (black) for a moment when the measured wind is 38 knots along 160°N (left), and compares spatial profiles of a turn under the two wind regimes; (c) compares the commanded airspeed (left) and commanded crab angle (right) for the two wind regimes.

Chapter 7

Discussion and Conclusion

We have presented κITE , a decoupled trajectory optimization approach for UAVs that computes feasible, time-optimal trajectories while explicitly accounting for disturbance due to wind. We have also described our approaches for takeoff and landing, resulting in a complete system that generates trajectories from start to goal. To our knowledge, there is no other existing approach that demonstrably optimizes for path and velocity for UAVs in the presence of wind, and generates smooth, dynamically feasible trajectories in near real-time. Our main conclusions can be summarized as follows-

1. Our algorithm effectively plans smooth, dynamically-feasible trajectories that explicitly account for wind. It is near real-time, which is essential for practical deployment on aerial vehicles such as our autonomous helicopter.
2. Decoupling path optimization from velocity optimization is an extremely effective strategy to make the highly non-convex problem tractable. Our derivation of appropriate curvature limits, taking into account all appropriate dynamic limits of the system (roll and its derivatives, airspeed and acceleration) is the key ingredient that allows us to decouple path from velocity in a principled manner. This is an especially difficult problem given the presence of wind, which automatically introduces time-dependence even in the path optimization stage.
3. Searching for arcs in the curvature space allows us to directly address non-trivial steering constraints such as roll-rate. The curvature space also provides an elegant mathematical representation of the steering control input.
4. We have demonstrated the effectiveness of our algorithm with over 20 successful real-world flight tests using both onboard wind measurements and estimates from weather stations, and analyzed the 3 cases where it failed to find a feasible trajectory.
5. κITE also determines whether it is safe to takeoff/land in the current wind regime, and aborts before takeoff if the conditions are adverse to safe flight.
6. While we have not currently tested with very long missions ($\geq 100km$), our simulation results indicate that κITE scales gracefully with mission length and the number of waypoints. For example, κITE computes a trajectory for a $\sim 500km$ mission in $\sim 7s$.
7. Our emphasis on treating roll and roll-rate as strict constraints arises out of our discus-

sion with helicopter pilots, as well as the need to ensure operation within control margins. However, pilots might sometimes violate these limits during actual flight if the situation demands, and determining the exact tradeoffs in a pilot's internal cost function using machine learning techniques (such as Inverse Reinforcement Learning) remains an exciting avenue for future work.

8. κITE is currently constrained to lie on the polyline defined by the mission waypoints. Including the problem of determining turn end-points in the optimization process is an area of future work. However, the algorithm includes logic that preprocesses the waypoints to make them suitable for the optimizer. For instance, extremely short segments and very sharp turns are replaced with dynamically feasible segments before being fed to the optimizer.
9. Another interesting area of future work is to use the motion primitives (turns and straight lines) that we generate in an RRT-style algorithm to relax the polyline assumption.
10. We have open-sourced a MATLAB implementation of κITE at `https://bitbucket.org/castacks/kite_optimizer`. κITE is an effective trajectory optimization technique for systems such as helicopters and fixed-wing aircraft that have steering and steering-rate constraints.

Appendix A

Curvature Bounds

A.1 Deriving bounds for curvature

The following subsections derive allowable bounds on curvature and curvature derivatives given dynamic limits.

We make use of the following relation between κ and ϕ:

$$\kappa = \frac{g \tan \phi}{v^2} \tag{A.1}$$

A.1.1 Time vs spatial derivatives

For a quantity r, we use the chain rule to specify the relation between time and spatial derivatives ($\dot{r}$ and r' respectively, and $v = \frac{ds}{dt} =$ speed):

$$r' = \frac{dr}{ds} = \frac{\dot{r}}{v} \tag{A.2}$$

A.1.2 Curvature bound

$$\kappa_{max} = \frac{g \tan \phi_{\max}}{v_{\max}} \tag{A.3}$$

A.1.3 Curvature derivative bound

Differentiating (A.1) with respect to arclength s, we get

$$\kappa' = \frac{g \left(\sec^2 \phi\right) \dot{\phi}}{v^3} - \frac{2\kappa a}{v^2}$$

Let $\alpha = g \left(\sec^2 \phi\right) \dot{\phi}$ and $\beta = 2\kappa a$. Therefore,

$$\kappa' = \frac{\alpha}{v^3} - \frac{\beta}{v^2}$$
$$\implies \frac{d\kappa'}{dv} = \frac{-3\alpha}{v^4} + \frac{2\beta}{v^3}$$

Setting this to zero, we get

$$v^* = \frac{3\alpha}{2\beta} = \gamma$$

Differentiating again w.r.t v, we get

$$\begin{aligned}\frac{d\kappa''}{dv} &= \frac{12\alpha}{v^5} - \frac{6\beta}{v} \\ &= \frac{6}{v^4}\left(\frac{2\alpha}{v} - \beta\right) \\ &= \frac{6}{v^4}\left(\frac{2\alpha}{\frac{3\alpha}{2\beta}} - \beta\right) \\ &= \frac{2\beta}{v^4}\end{aligned}$$

Since we want to derive a worst-case upper bound, we set $\beta = 2\kappa_{max}a_{\max}$. Therefore $\frac{d\kappa''}{dv} > 0$ at $v^* = \frac{3\alpha}{2\beta} = \gamma$. Similarly, we set $\alpha = g\dot{\phi}_{\max}$ for a conservative bound. The curvature-rate limit is now

$$\kappa'_{max} = \frac{\alpha}{\gamma^3} - \frac{\beta}{\gamma^2} \tag{A.4}$$

$$\left(\alpha = g\dot{\phi}_{\max}, \quad \beta = 2\kappa_{max}a_{\max}, \quad \gamma = \frac{3\alpha}{2\beta}\right) \tag{A.5}$$

A.1.4 Curvature double-derivative bound

For a conservative bound, we get

$$\kappa''_{max} = \frac{g\ddot{\phi}_{\max}}{v^4_{\max}} \tag{A.6}$$

Bibliography

[1] Daniel Althoff, Matthias Althoff, and Sebastian Scherer. Online safety verification of trajectories for unmanned flight with offline computed robust invariant sets. In *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, pages 3470–3477. IEEE, 2015. 1

[2] Erik P. Anderson, Randal W. Beard, and Timothy W. McLain. Real-time dynamic trajectory smoothing for unmanned air vehicles. *IEEE Trans. Contr. Sys. Techn.*, 13:471–477, 2005. 2.2

[3] Efstathios Bakolas and Panagiotis Tsiotras. Time-optimal synthesis for the zermelo–markov–dubins problem: the constant wind case. In *American Control Conference (ACC)*, pages 6163–6168, 2010. 2.2

[4] Efstathios Bakolas and Panagiotis Tsiotras. Optimal synthesis of the zermelo–markov–dubins problem in a constant drift field. *Journal of Optimization Theory and Applications*, 156(2):469–492, 2013. 2.2, 2.2

[5] James E Bobrow, Steven Dubowsky, and JS Gibson. Time-optimal control of robotic manipulators along specified paths. *The international journal of robotics research*, 4(3):3–17, 1985. 2.2, 5.2

[6] Howie M Choset. *Principles of robot motion: theory, algorithms, and implementation*. MIT press, 2005. 2.2, 5.2

[7] Sanjiban Choudhury , Sankalp Arora, and Sebastian Scherer. The planner ensemble and trajectory executive: A high performance motion planning system with guaranteed safety. In *AHS 70th Annual Forum, Montre al, Que bec, Canada*, May 2014. 6.2.1

[8] Sanjiban Choudhury and Sebastian Scherer. The dynamics projection filter (dpf)-real-time nonlinear trajectory optimization using projection operators. In *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, pages 644–649. IEEE, 2015. 2.2

[9] Sanjiban Choudhury, Sankalp Arora, and Sebastian Scherer. The planner ensemble: Motion planning by executing diverse algorithms. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2389–2395. IEEE, 2015. 1

[10] Vishal Dugar , Sanjiban Choudhury, and Sebastian Scherer. A kite in the wind: Smooth trajectory optimization in a moving reference frame. In *IEEE International Conference on Robotics and Automation*, Singapore, May 2017. 1

[11] Vishal Dugar , Sanjiban Choudhury, and Sebastian Scherer. Smooth trajectory optimization

in wind: First results on a full-scale helicopter. In *AHS International 73rd Annual Forum, Forth Worth, Texas, USA*, May 2017. 1

[12] FAA. *Helicopter Flying Handbook*. 1, 3

[13] Thierry Fraichard and Alexis Scheuer. From reeds and shepp's to continuous-curvature paths. *IEEE Transactions on Robotics*, 20(6):1025–1035, 2004. 2.2

[14] Emilio Frazzoli, Munther A Dahleh, and Eric Feron. Real-time motion planning for agile autonomous vehicles. *Journal of Guidance, Control, and Dynamics*, 25(1):116–129, 2002. 2.2, 2.1

[15] Chad Goerzen, Zhaodan Kong, and Bernard Mettler. A survey of motion planning algorithms from the perspective of autonomous uav guidance. *Journal of Intelligent and Robotic Systems*, 57(1-4):65–100, 2010. 1

[16] Jeong hwan Jeon, Raghvendra V Cowlagi, Steven C Peters, Sertac Karaman, Emilio Frazzoli, Panagiotis Tsiotras, and Karl Iagnemma. Optimal motion planning with the half-car dynamical model for autonomous high-speed driving. In *2013 American Control Conference*, pages 188–193. IEEE, 2013. 2.2, 2.1, 5.2

[17] Dongwon Jung and Panagiotis Tsiotras. On-line path generation for small unmanned aerial vehicles using b-spline path templates. In *AIAA Guidance, Navigation and Control Conference, AIAA*, volume 7135, 2008. 2.2

[18] Mrinal Kalakrishnan, Sachin Chitta, Evangelos Theodorou, Peter Pastor, and Stefan Schaal. Stomp: Stochastic trajectory optimization for motion planning. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 4569–4574. IEEE, 2011. 2.2, 2.1

[19] Sertac Karaman and Emilio Frazzoli. Incremental sampling-based algorithms for optimal motion planning. *Robotics Science and Systems VI*, 104, 2010. 2.2, 2.1

[20] Alonzo Kelly and Bryan Nagy. Reactive nonholonomic trajectory generation via parametric optimal control. *The International Journal of Robotics Research*, 22(7-8):583–601, 2003. 2.2, 5.3

[21] Steven M LaValle. *Planning algorithms*. Cambridge university press, 2006. 2.1

[22] Steven M LaValle and James J Kuffner Jr. Randomized kinodynamic planning. *The International Journal of Robotics Research*, 20(5):378–400, 2001. 2.1

[23] Maxim Likhachev and Dave Ferguson. Planning long dynamically feasible maneuvers for autonomous vehicles. *The International Journal of Robotics Research*, 28(8):933–945, 2009. 2.2, 2.1

[24] Thomas Lipp and Stephen Boyd. Minimum-time speed optimisation over a fixed path. *International Journal of Control*, 87(6):1297–1311, 2014. 2.2

[25] Brian MacAllister, Jonathan Butzke, Alex Kushleyev, Harsh Pandey, and Maxim Likhachev. Path planning for non-circular micro aerial vehicles in constrained environments. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 3933–3940. IEEE, 2013. 2.2, 2.1

[26] Timothy G McGee, Stephen Spry, and J Karl Hedrick. Optimal path planning in a constant

wind with a bounded turning rate. In *AIAA Guidance, Navigation, and Control Conference and Exhibit*, pages 1–11. Reston, VA, 2005. 1, 2.2, 2.2

[27] Michael Otte, William Silva, and Eric Frew. Any-time path-planning: Time-varying wind field + moving obstacles. In *IEEE International Conference on Robotics and Automation*, Stockholm, Sweden, 2016. 1

[28] Mihail Pivtoraiko, Ross A Knepper, and Alonzo Kelly. Differentially constrained mobile robot motion planning in state lattices. *Journal of Field Robotics*, 26(3):308–333, 2009. 2.2, 2.1

[29] Raymond W Prouty. *Helicopter performance, stability, and control*. 1995. 1

[30] Nathan Ratliff, Matt Zucker, J Andrew Bagnell, and Siddhartha Srinivasa. Chomp: Gradient optimization techniques for efficient motion planning. In *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*, pages 489–494. IEEE, 2009. 2.2, 2.1

[31] Alexis Scheuer and Christian Laugier. Planning sub-optimal and continuous-curvature paths for car-like robots. In *Intelligent Robots and Systems, 1998. Proceedings., 1998 IEEE/RSJ International Conference on*, volume 1, pages 25–31. IEEE, 1998. 2.2

[32] John Schulman, Jonathan Ho, Alex X Lee, Ibrahim Awwal, Henry Bradlow, and Pieter Abbeel. Finding locally optimal, collision-free trajectories with sequential convex optimization. In *Robotics: science and systems*, volume 9, pages 1–10, 2013. 2.2, 2.1

[33] Martin Seleck, Petr Vana, Milan Rollo, and Tomáš Meiser. Wind corrections in flight path planning. *Int J Adv Robotic Sy*, 10(248), 2013. 1

[34] Laszlo Techy. Optimal navigation in planar time-varying flow: Zermelos problem revisited. *Intelligent Service Robotics*, 4(4):271–283, 2011. 1, 2.2

[35] Laszlo Techy, Craig A Woolsey, and Kristi A Morgansen. Planar path planning for flight vehicles in wind with turn rate and acceleration bounds. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 3240–3245. IEEE, 2010. 2.2, 2.2

[36] Diederik Verscheure, Bram Demeulenaere, Jan Swevers, Joris De Schutter, and Moritz Diehl. Time-optimal path tracking for robots: A convex optimization approach. *IEEE Transactions on Automatic Control*, 54(10):2318–2327, 2009. 2.2

[37] Chanyeol Yoo, Robert Fitch, and Salah Sukkarieh. Online task planning and control for fuel-constrained aerial robots in wind fields. *The International Journal of Robotics Research*, page 0278364915595278, 2015. 1