A κITE in the Wind: Smooth Trajectory Optimization in a Moving Reference Frame

Vishal Dugar[1], Sanjiban Choudhury[1] and Sebastian Scherer[1]

***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 paper, we address the problem of planning long, dynamically feasible, time-optimal trajectories in the presence of wind (which creates a moving reference frame).**

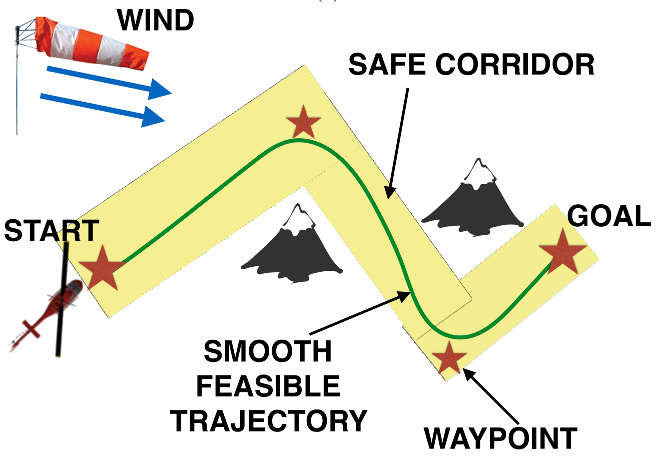
We present an algorithm, κITE , that elegantly decouples the joint trajectory optimization problem into individual path optimization in a fixed ground frame and a velocity profile optimization in a moving reference frame. The key idea is to derive a decoupling framework that guarantees feasibility of the final fused trajectory. Our results show that κITE is able to produce high-quality solutions for planning with a 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.

I. Introduction

There has recently been extensive research on unmanned aerial vehicles (UAVs) such as helicopters and fixed-wing aircraft that can travel large distances [1], [19], [8], [25], [12]. 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 [20], [10]. In conjunction with these requirements, these systems must be cognizant of the effect of wind on flight profiles [10], [21], [18], [22]. 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 as shown in Fig. 1b.

Consider the problems faced 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 planning is done in the ground frame, the dynamics constraints are no longer stationary and vary along the path. In addition to wind, the other main challenges are satisfying non-holonomic constraints due to vehicle dynamics and scaling to large distances. This results in a complex multi-resolution, non-convex planning problem. Finally, the optimizer is required

(a)



(b)

Fig. 1: The trajectory optimization problem given corridor constraints, dynamic limits and wind. (a) Our full-size helicopter platform.

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), 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 pa-

* This work would not have been possible without the dedicated efforts of the entire AACUS TALOS team and was supported by ONR under contract N00014-12-C-0671. Approved for Public Release; DCN No. 43-2249-16

[1] V. Dugar, S. Choudhury and S. Scherer are with The Robotics Institute at Carnegie Mellon University, Pittsburgh, Pennsylvania, USA. Email: {vdugar, sanjibac, basti}@andrew.cmu.edu

rameterization 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.

The paper is organized as follows - we explore related work and precisely define the problem. We then describe our approach, and show results from both simulations and experiments on a full-size helicopter. We conclude with a discussion of our work.

II. Related Work

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. [14] proposes an online, corridor-constrained smoothing algorithm that uses B-spline templates to generate paths, but not time profiles. Sampling-based techniques like [11], [15] 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], [22] and [4] to characterize optimal solutions, but they use sharp turns and constant speed, neither of which are practical. [18] also uses a bounded turning radius assumption to yield minimum-time trajectories with constant speed. [23] 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. 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], [13]). We use concepts from previous work done on optimizing velocity profiles given a fixed path and a finite set of velocity bottlenecks ([17], [24]) to compute time-optimal velocity profiles.

III. Problem Definition

In this section, we introduce the trajectory optimization problem we wish to solve. We first provide some context into the desired nature of the optimizer. The motion planning system typically consists of two modules - the global planner and the local planner. The global planner is responsible for producing a nominal trajectory that has guarantees with respect to the overall mission such as ensuring the vehicle can stay in a designated safe flight corridor, respect speed limits along segments and can feasibly transition between corridors in presence of wind. The local planner is responsible for locally (few kilometers) repairing the trajectory to avoid sensed obstacles. Since the global planner's trajectory is executed under nominal circumstances, it is imperative that the trajectory be of high quality in terms of time and feasibility. The local planner can sacrifice quality to ensure safety of the aircraft. In this work, we focus on the former and refer the reader to [7] for the latter.

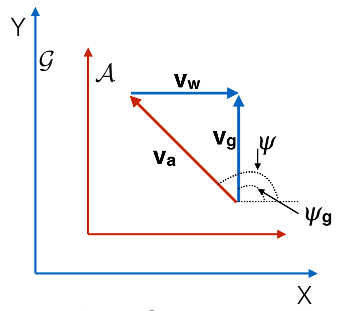


Fig. 2: The fixed ground frame $\mathcal{G}$ and moving reference frame $\mathcal{A}$. The velocity vector in airframe v is added to the windspeed vector v_w to produce a velocity vector in groundframe v_g. The heading in $\mathcal{G}$ is ψ_g while in airframe is ψ

A. Notation

TABLE I: 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

B. Dynamics of a Fixed Wing UAV in Wind

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. 2. 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-

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{v} \\ \dot{a} \\ \dot{\psi} \\ \dot{\phi} \\ \dot{\omega} \end{bmatrix} = \begin{bmatrix} v\cos\psi \\ v\sin\psi \\ a \\ j \\ \frac{g\tan\phi}{\sqrt{\dot{x}^2+\dot{y}^2}} \\ \omega \\ \alpha \end{bmatrix} \tag{1}$$

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 (1) and limits are translated into higher order constraints and bounds on $\sigma(t)$. Without loss of generality, we assume the 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-

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

C. Input Mission

Let V_{start} and V_{goal} be the specified start and goal velocities. The input mission consists of N waypoints $\{w_1, \ldots, w_N\}$. These waypoints define $N-1$ segments. A 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. A maximum segment velocity is specified for each segment $V_{\mathrm{st},i}$. If a configuration belongs to a corridor, it must satisfy the segment velocity limit. The corridors are either specified by a human, or are determined by a route-planning module that is invoked prior to calling κITE .

D. Trajectory Optimization Problem

$$\begin{aligned}
\min_{\sigma(.),t_f} \quad & t_f \\
\text{s.t} \quad & \left.\begin{aligned}
\sqrt{\dot{x}^2(t) + \dot{y}^2(t)} &\geq v_{\min} \\
\sqrt{\dot{x}^2(t) + \dot{y}^2(t)} &\leq v_{\max} \\
\sqrt{\ddot{x}^2(t) + \ddot{y}^2(t)} &\leq a_{\max} \\
\sqrt{\dddot{x}^2(t) + \dddot{y}^2(t)} &\leq j_{\max} \\
|\phi(t)| &\leq \phi_{\max} \\
\left|\dot{\phi}(t)\right| &\leq \dot{\phi}_{\max} \\
\left|\ddot{\phi}(t)\right| &\leq \ddot{\phi}_{\max}
\end{aligned}\right\} \begin{array}{l}\text{Derivative}\\ \text{Bounds}\end{array} \\
& \left.\begin{aligned}
\psi(t) &= \tan^{-1}\left(\frac{\dot{y}(t)}{\dot{x}(t)}\right) \\
\dot{\psi}(t) &= \frac{g\tan\phi}{\sqrt{\dot{x}^2(t) + \dot{y}^2(t)}}
\end{aligned}\right\} \begin{array}{l}\text{Dynamics}\\ \text{Constraints}\end{array} \\
& \left.\begin{aligned}
\sqrt{\dot{x}^2(0) + \dot{y}^2(0)} &= V_{start} \\
\sqrt{\dot{x}^2(t_f) + \dot{y}^2(t_f)} &= V_{goal} \\
\sum_{i=1}^{N-1} \mathbb{I}_i\left(\texttt{Proj}(\sigma(t), v_w)\right) &> 0 \\
\left(\sqrt{\dot{x}^2(t) + \dot{y}^2(t)}\right)\mathbb{I}_i(\sigma(t)) &\leq V_{\mathrm{st},i}, \\
\text{for } i \in \{1, \ldots, N\}&
\end{aligned}\right\} \begin{array}{l}\text{Route}\\ \text{Constraints}\end{array} \\
& \text{for } t \in [0, t_f]
\end{aligned} \tag{3}$$

IV. Approach

To make the solution-search tractable, we decouple the optimization problem into a path optimization and a velocity profile optimization ([6], [5], [13]). The optimization approach, summarized in Fig 3 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 $V_{\mathrm{lim},i}$, such that the section is feasible for any velocity profile limited by $V_{\mathrm{lim},i}$. In the interest of time-optimality, the objective of this optimizer is to maximize the velocity limit $V_{\mathrm{lim},i}$ while keeping the total arclength of the section small.
2) Phase B: *Velocity Optimization*. This phase optimizes velocity at specific *control points* 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.

A. Phase A: Path Optimization

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.

1) 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 $\{\xi^1_{\mathrm{st}}, \xi^1_{\mathrm{arc}}, \xi^2_{\mathrm{st}}, \xi^2_{\mathrm{arc}} \ldots\}$. 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.

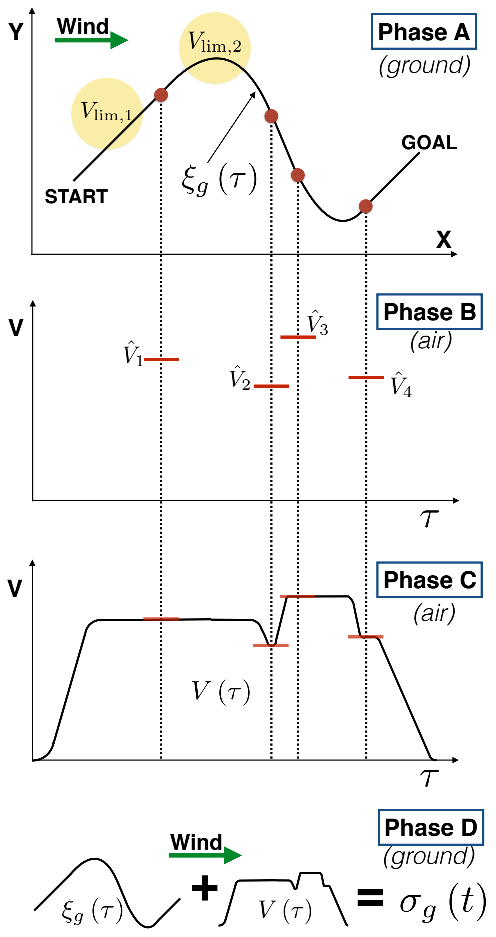


Fig. 3: Overview of κITE

2) 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 $\texttt{CurvLimit}\left(v_{\max}, \phi_{\max}, \dot{\phi}_{\max}, \ddot{\phi}_{\max}, a_{\max}\right)$ (proof at [9]):

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

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. To convert between ψ_g and ψ, we use the function $\texttt{HeadingInAir}\,(\psi_g, v, v_w)$:

$$\psi = \operatorname{acos}\left(-\frac{v_w \sin(\psi_g)}{v}\right) - \frac{\pi}{2} + \psi_g \tag{5}$$

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

3) Arc Parameterization: Let s be the arc-length of a path. Let $\xi(s)$ be an arclength parameterized path. Building on [16], each arc ξ^i_{arc} is represented as a C1 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^1_f, S^2_f, S^3_f denote the arclengths of the three sections, and S_f denotes the total arclength. 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 $\texttt{CurvPolyGnd}\,(\kappa(s))$:

$$\begin{aligned}
\psi(s) &= \int_0^s \kappa(s) \\
x(s) &= \int_0^s cos(\psi(s)) + \frac{v_w}{V_{\text{lim},i}} \int_0^s ds \\
y(s) &= \int_0^s sin(\psi(s)) \\
\psi_g(s) &= \tan^{-1}\left(\frac{\dot{y}(s)}{\dot{x}(s)}\right)
\end{aligned} \tag{6}$$

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 $\texttt{CurvPoly}\,(\kappa_0, \kappa_f, S_f, \kappa_{max}, \kappa'_{max}, \kappa''_{max})$:

$$\begin{aligned}
\text{find} \quad & \kappa(s) \\
& \kappa(0) = \kappa_0, \kappa(S_f) = \kappa_f \\
& \kappa'(0) = 0, \kappa'(S_f) = 0 \\
& \kappa(s) \le \kappa_{max}, \kappa'(s) \le \kappa'_{max}, \kappa''(s) \le \kappa''_{max}
\end{aligned} \tag{7}$$

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.

4) 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 $V_{\text{lim}} = \left[V_{\text{st},1} \quad V_{\text{arc},1} \quad \dots \quad V_{\text{arc},N-2} \quad V_{\text{st},N-1}\right]$, that are used by Phase B.

B. Phase B: Time Optimization

1) 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

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

```
1  for $v \leftarrow [v_{\max}, v_{\min}]$ do
2      $(\kappa_{max}, \kappa'_{max}, \kappa''_{max}) \leftarrow$ CurvLimit $\left(v, \phi_{\max}, \dot{\phi}_{\max}, \ddot{\phi}_{\max}, a_{\max}\right)$
3      $\Delta\psi \leftarrow$ HeadingInAir $(\angle(w_{i+1}, w_{i+2})) -$ HeadingInAir $(\angle(w_i, w_{i+1}))$
4      for $\kappa \leftarrow [\kappa_{min}, \kappa_{max}]$ do
5          $\kappa_2(s) = \kappa$
6          for $S_f^1 \leftarrow [S_f^{\min}, S_f^{\max}]$ do
7              $\kappa_1(s) \leftarrow$ CurvPoly $\left(0, \kappa, S_f^1, \kappa_{max}, \kappa'_{max}, \kappa''_{max}\right)$
8              if $\kappa_1(s) \in \emptyset$ then
9                  break
10         $S_f^3 \leftarrow S_f^1$
11         $\kappa_3(s) \leftarrow$ CurvPoly $\left(\kappa, 0, S_f^3, \kappa_{max}, \kappa'_{max}, \kappa''_{max}\right)$
12         $S_f^2 = \frac{\Delta\psi - \int_0^{S_f^1} \kappa_1(s) - \int_0^{S_f^3} \kappa_3(s)}{\kappa}$
13         if $S_f^2 \geq 0$ then
14             break
15     $\kappa(s) \leftarrow \begin{bmatrix}\kappa_1(s) & \kappa_2(s) & \kappa_3(s)\end{bmatrix}$
16     $\xi^i_{\text{arc,gnd}}(s) \leftarrow$ CurvPolyGnd $(\kappa(s))$
17     if $\sum_{j=i}^{i+1} \mathbb{I}_j \left(\xi^i_{\text{arc,gnd}}(s)\right) > 0$ then
18         break
19 return $v, \xi^i_{\text{arc}}(s), \xi^i_{\text{arc,gnd}}(s)$
```

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 $\hat{V}_0 = V_{start}, \hat{V}_{N+1} = V_{goal}$):

$$\begin{aligned} \underset{\{\hat{V}_i\}}{\text{minimize}} \quad & t_f\left(\{\hat{V}_i\}, \{S_i\}, \hat{a}_{\max}\right) \\ \text{subject to} \quad & \hat{V}_i \leq V_{\text{lim},i} \\ & \frac{|\hat{V}^2_{i+1} - \hat{V}^2_i|}{2\hat{a}_{\max}} \leq S_i \end{aligned} \tag{8}$$

The total time t_f is defined as follows-

$$t_f = \sum_i t_i \tag{9}$$

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

$$V_{mid} = \min\left(\sqrt{\frac{2\hat{a}_{\max}S_i + \hat{V}_i^2 + \hat{V}_{i+1}^2}{2}}, V_{\text{lim},i}\right) \tag{10}$$

$$t_i = \frac{2V_{mid} - \hat{V}_{i+1} - \hat{V}_i}{\hat{a}_{\max}} + \tag{11}$$

$$\left(S_i - \frac{2V^2_{mid} - \hat{V}^2_{i+1} - \hat{V}^2_i}{2\hat{a}_{\max}}\right)\frac{1}{V_{mid}} \tag{12}$$

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

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

2) 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:

$$\hat{V}_2 = \sqrt{\hat{V}_1^2 + 2aS} \tag{13}$$

C. 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. 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 $V(t)$, $t \in [0, t_f]$. 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.

D. 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:

$$v_g(\tau) = \sqrt{V(\tau)^2 - v_w^2\sin^2(\psi_g(\tau))} + v_w\cos(\psi_g(\tau)) \tag{14}$$

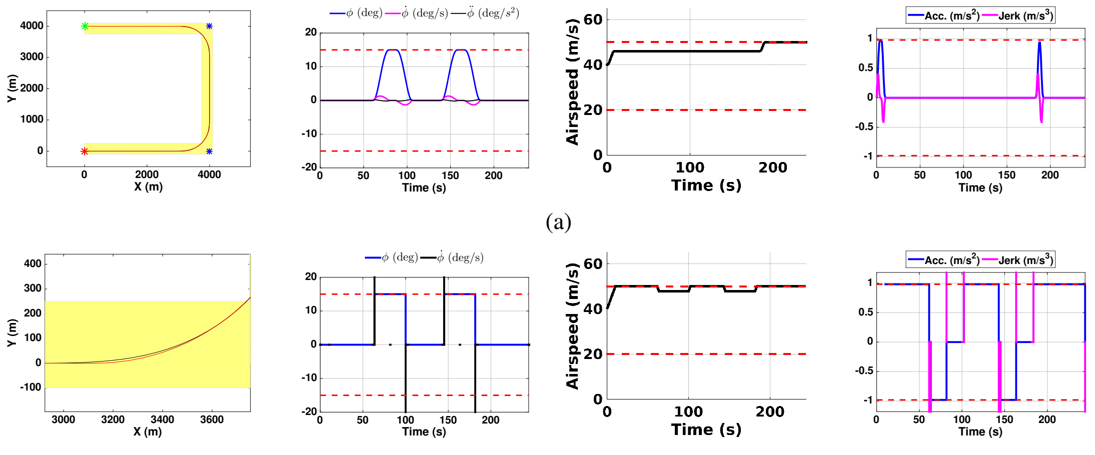


Fig. 4: Comparing the spatial, roll, speed and acceleration profiles of κITE (top) with the baseline (bottom). Corridors are highlighted in yellow, and limits are represented by red lines. A blowup of (bottom left) the spatial profiles (top left) shows κITE in black and the baseline in red.

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

$$S_g(\tau) = S_g(x(\tau)) = \int_0^{x(\tau)} \sqrt{1+y'(x)^2}dx, \quad y' = \frac{dy}{dx} \tag{15}$$

3) Compute the time profile:

$$t(\tau) = \int_0^{S_g(\tau)} \frac{dS_g}{v_g(\tau)} \tag{16}$$

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

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

$$\psi(t) = \text{atan}\left(\frac{v_g(t)\sin(\psi_g(t))}{v_g(t)\cos(\psi_g(t)) - v_w}\right) \tag{17}$$

5) Compute the roll profile:

$$\phi_g(t) = \phi(t) = \text{atan}\left(\frac{V(t)\dot{\psi}(t)}{g}\right) \tag{18}$$

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

V. Results

A. Implementation

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.

B. Solution Quality

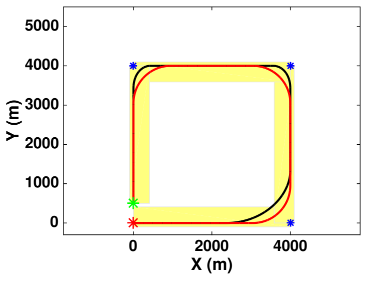
1) **Respecting dynamic limits**
Fig. 4 shows how κITE is able to respect limits on speed, acceleration, jerk, roll, roll-rate and roll-racceleration. This is essential for stable trajectory tracking, especially in the presence of disturbing forces such as wind that might cause the system to exceed its control margin and enter into a potentially hazardous state. Fig. 4 compares the output of κITE with the baseline.

2) **Performance With Wind**
Fig. 5 shows a scenario where a helicopter is flying in the presence of a $20m/s$ wind directed along the $+ve$ X-axis. As Fig. 5(b) shows, a feedback controller that attempts to follow the trajectory computed without taking wind into account (red trajectory in Fig. 5(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. 5(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.

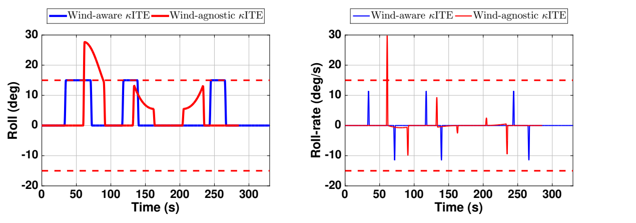
C. Scalability and versatility

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 shows one such situation, in which κITE computes a $\sim$ 290 km with a mixture of long and short segments.

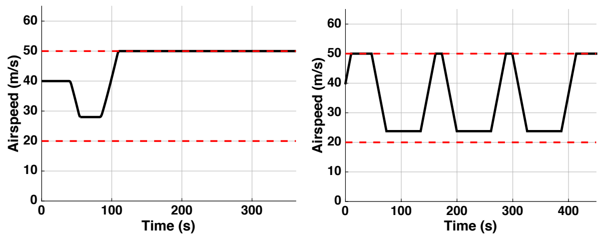
We also tested a $C++$ implementation of κITE on 100 randomly generated problems across three runs with 10, 25 and 50 waypoints respectively. Individual segment lengths range from $300m$ to $5000m$, and are randomly chosen for each waypoint, as are the angles between segments. A pre-computed lookup table 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 II) :



(a)



(b)



(c)

Fig. 5: 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.

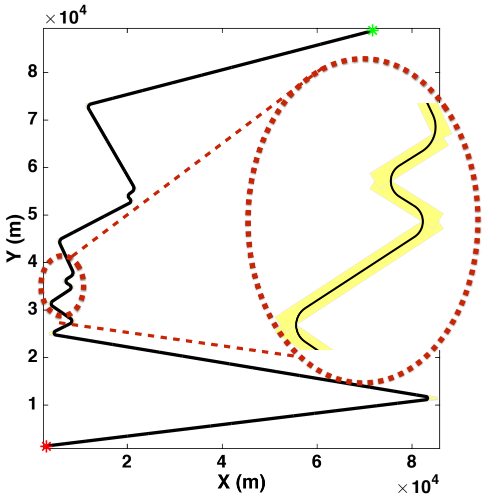
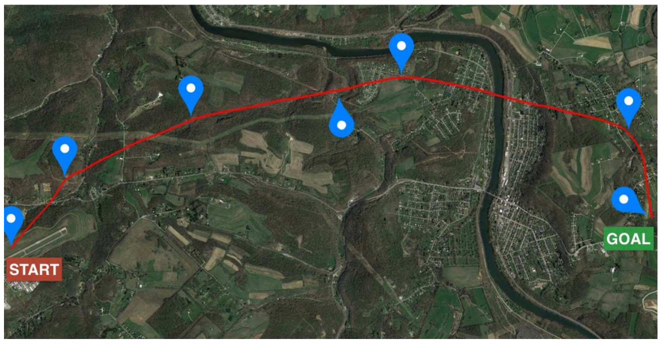
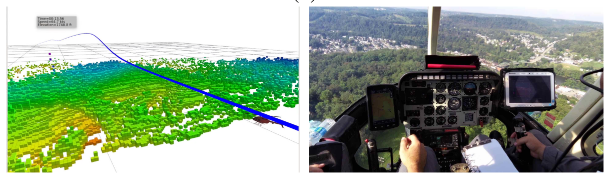


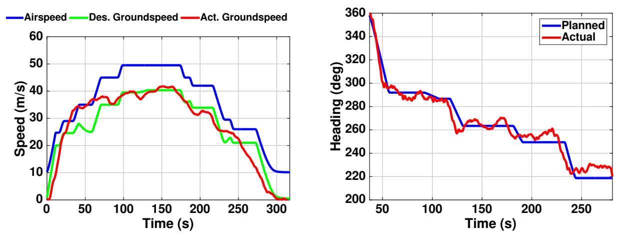
Fig. 6: A $\sim$ 290 km trajectory computed by κITE . The problem has both long route segments and short segments with closely spaced turns, as the cutout shows. κITE 's solution structure allows it to efficiently deal with such problems. Again, the corridor is highlighted in yellow.



(a)



(b)



(c)

Fig. 7: Results from a test conducted on a full-size helicopter with a $\sim 10m/s$ wind blowing towards 110°N. (a) shows the complete trajectory along with the waypoints; (b) shows a simulation of the evidence grid, and a view from the cockpit; (c) shows tracking performance for groundspeed and ground-frame heading, along with the desired airspeed. This trajectory was computed in real-time onboard the helicopter. κITE serves as a global planner in our planning architecture, and reasons about the entire mission at once. It feeds into a local planner that accounts for contingencies like sensed obstacles.

TABLE II: 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

D. *Real-world experiments*

κITE currently runs on our autonomous full-scale helicopter, where it re-computes trajectories online whenever the wind conditions (sensed with a pitot tube onboard) change. It functions as a global planner in our C++ architecture, and feeds into a local planner which deals with real-time obstacle avoidance. Fig. 7 shows results from one of our flight tests, where the system was able to track the commanded trajectory while remaining within its control margins at all times, in the presence of a $\sim 10m/s$ wind blowing towards 110°N (measured with the onboard pitot tube). Non-smooth, wind-agnostic trajectories that violate dynamic limits are hazardous for such systems, and necessitate the use of a method that guarantees nominal feasibility. So far, we have successfully tested κITE in 23 flights with winds up to $20m/s$.

VI. Discussion

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. Our path optimizer (Phase A) generates paths that satisfy corridor constraints in the presence of wind while executing the smoothest possible turns (i.e. turns with minimum curvature) at the maximum possible airspeeds. These paths are guaranteed to be dynamically feasible after subsequent velocity optimization, which makes the decoupling possible in the first place. We then formulate and solve a nonlinear velocity optimization problem (Phase B) that minimizes time while scheduling feasible control-point velocities. Since the number of control points are limited ($2N-4$), this allows us to quickly compute a time-optimal, pointwise velocity profile. We then generate a smooth, jerk-limited velocity profile for the entire path (Phase C). Our final stage combines the path and velocity profile to produce the final time-parameterized trajectory (Phase D).

While we have assumed a constant wind for the sake of presentation, it is trivial to extend the system to more realistic scenarios where the wind varies along different segments of the mission, as long as the wind estimate varies smoothly. The entire optimization process is near real-time. We have demonstrated the robustness and quality of our approach with both simulation results and real-world experiments on a full-size helicopter, and κITE is currently implemented in its onboard planning architecture. 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. Finally, we have also released open-source MATLAB code for the optimizer.

κITE is currently constrained to lie close to the poly-line defined by the mission waypoints. A future area of work deals with cases where the aircraft is only required to stay within the safe corridor and not along the given poly-line. It would be appropriate to solve for optimal turn end-points to maximize time-efficiency in such cases.

References

[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.

[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.

[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.

[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.

[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.

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

[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.

[8] 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.

[9] Vishal Dugar. Additional proofs. https://goo.gl/AP3OG5.

[10] FAA. *Helicopter Flying Handbook*.

[11] 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.

[12] 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.

[13] 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.

[14] 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.

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

[16] 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.

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

[18] 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.

[19] 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.

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

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

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

[23] 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.

[24] 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.

[25] 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.