Smooth Trajectory Optimization in Wind: First Results on a Full-Scale Helicopter

Vishal Dugar
MS Student
Robotics Institute, Carnegie Mellon University
Pittsburgh, PA, USA

Sanjiban Choudhury
PhD Student
Robotics Institute, Carnegie Mellon University
Pittsburgh, PA, USA

Sebastian Scherer
Systems Scientist
Robotics Institute, Carnegie Mellon University
Pittsburgh, PA, USA

ABSTRACT

A significant challenge for unmanned aerial vehicles is flying long distances in the presence of wind. The presence of wind, which acts like a forcing function on the system dynamics, significantly affects control authority and flight times. While there is a large body of work on the individual topics of planning long missions and path planning in wind fields, these methods do not scale to solve the combined problem under real-time constraints. In this paper, we address the problem of planning long, dynamically feasible, time-optimal trajectories in the presence of wind for a full-scale helicopter. We build on our existing algorithm, κITE , which accounts for wind in a principled and elegant way, and produces dynamically-feasible trajectories that are guaranteed to be safe in near real-time. It uses a novel framework to decouple path optimization in a fixed ground frame from velocity optimization in a moving air frame. We present extensive experimental evaluation of κITE on an autonomous helicopter platform (with a human safety pilot in the loop) with data from over 23 missions in winds up to $20m/s$ and airspeeds up to $50m/s$. Our results not only shows the efficacy of the algorithm and its implementation, but also provide insights into failure cases that we encountered. This paves the way forward for autonomous systems to exhibit pilot-like behavior when flying missions in winds aloft.

NOTATION

We make use of the following notation in this work. The subscript g is used to indicate the ground-frame equivalent of the trajectory variables defined below.

Table 1: Notation

Symbol	Description
x	x-coord in air frame
y	y-coord in air frame
ϕ	roll in air frame
ψ	yaw in air frame
v	airspeed
a	acc in air frame
j	jerk in air frame
σ	traj in air frame
ξ	path in air frame
s	arc distance
τ	index

Fig. 1: A modified Bell 206 with onboard computation serves as our full-size autonomous helicopter platform.

INTRODUCTION

Autonomous, full-size helicopters are increasingly required to fly long missions in the presence of wind. Missions can be $\approx 500km$ long, with the helicopter flying at airspeeds of $50m/s$ (Refs. 1–5). The success of these missions depends on the ability of the helicopter to plan smooth flight-profiles that are time-optimal, respect constraints such as dynamics

Presented at the AHS International 73rd Annual Forum & Technology Display, Fort Worth, Texas, USA, May 9–11, 2017.
Copyright © 2017 by AHS International, Inc. All rights reserved.

and safety envelopes, and additionally explicitly account for disturbances caused due to wind (which results in a moving reference frame). We will explain these requirements with an anecdotal example.

Consider a full-scale helicopter as shown in Fig. 1 flying a long distance mission between two airports. A ground station specifies a *flight corridor* that keeps the system out of restricted airspace (Fig. 4). The system has to plan a trajectory that stays within this flight corridor. Firstly, the trajectory has to satisfy the dynamic constraints on the helicopter's pose and derivatives of the pose (say, bank and bank-rate). Secondly, the trajectories are incentivized to be smooth. A primary reason for this is to emulate pilot-like behavior - we gathered from discussions with pilots that while executing turns, it is desirable to maintain small changes in bank angle and bank rates. Smooth trajectories also ensure that the helicopter operates well within its control margin, and can therefore respond to unforeseen circumstances without compromising on safety.

Thirdly, constraining the trajectory to lie inside the flight corridor is a major challenge in the presence of wind, which acts like a forcing function causing deviations from the intended ground track. The trajectory generation process, or planner, must explicitly account for this disturbance, or the aircraft might fly into potentially hazardous airspace. Reasoning about wind (a moving reference frame) also becomes important because the dynamic limits of helicopters (such as maximum airspeed, maximum bank rate and jerk) are specified in this moving airframe. Lastly, changing environmental conditions or mission requirements might necessitate replanning on the fly, requiring a near real-time trajectory planner. Hence, we tackle the problem of generating smooth, time-optimal trajectories for aircraft such as full-scale autonomous helicopters flying through a pre-defined safe corridor defined by waypoints, in the presence of wind.

In (Ref. 6), we presented an algorithm, κITE (Curvature (κ) parameterization Is very Time Efficient), to efficiently solve this optimization problem, and here we present experimental results from real-world flight tests. The key ideas behind the effectiveness of the algorithm are -

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.

RELATED WORK

The problem of generating feasible trajectories for UAVs has previously been explored in the literature. (Ref. 20) builds on

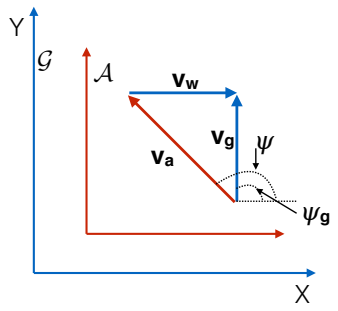


Fig. 2: The fixed ground frame $\mathscr{G}$ and moving reference frame $\mathscr{A}$. The velocity vector in airframe v is added to the wind-speed vector v_w to produce a velocity vector in groundframe v_g. The heading in $\mathscr{G}$ is ψ_g while in airframe is ψ

the Dubins solution and uses fixed-radius arcs to link straight segments with constant speed. (Ref. 21) proposes an online, corridor-constrained smoothing algorithm that uses B-spline templates to generate paths, but not time profiles. Sampling-based techniques like (Ref. 22), (Ref. 23) 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 (Ref. 18), (Ref. 17) and (Ref. 19) to characterize optimal solutions, but they use sharp turns and constant speed, neither of which are practical. (Ref. 16) also uses a bounded turning radius assumption to yield minimum-time trajectories with constant speed. (Ref. 24) 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 ((Ref. 7), (Ref. 8), (Ref. 9)). We use concepts from previous work done on optimizing velocity profiles given a fixed path and a finite set of velocity bottlenecks ((Ref. 25), (Ref. 26)) to compute time-optimal velocity profiles.

PROBLEM SETUP

In this section, we formally define the constrained trajectory optimization problem that we wish to solve. We reproduce material from (Ref. 6) here for the sake of completeness.

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 - $\mathscr{A}$ and $\mathscr{G}$ as shown in Fig. 2. The state space dynamics are defined in $\mathscr{A}$. A state space trajectory in $\mathscr{A}$ can be projected to $\mathscr{G}$ using the wind.

Let the $\mathbb{R}^7$ state space defined in airframe $\mathscr{A}$ 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 $\mathscr{A}$. 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-

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

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 $V_{\text{st},i}$. 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 .

Trajectory Optimization Problem

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

APPROACH

To make the solution-search tractable, we decouple the optimization problem into a path optimization and a velocity profile optimization ((Ref. 7), (Ref. 8), (Ref. 9)). The optimization approach, reproduced from (Ref. 6) and 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 $V_{\text{lim},i}$, such that the section is feasible for any velocity profile limited by $V_{\text{lim},i}$. In the interest of time-optimality, the objective of this optimizer is to maximize the velocity limit $V_{\text{lim},i}$ 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.

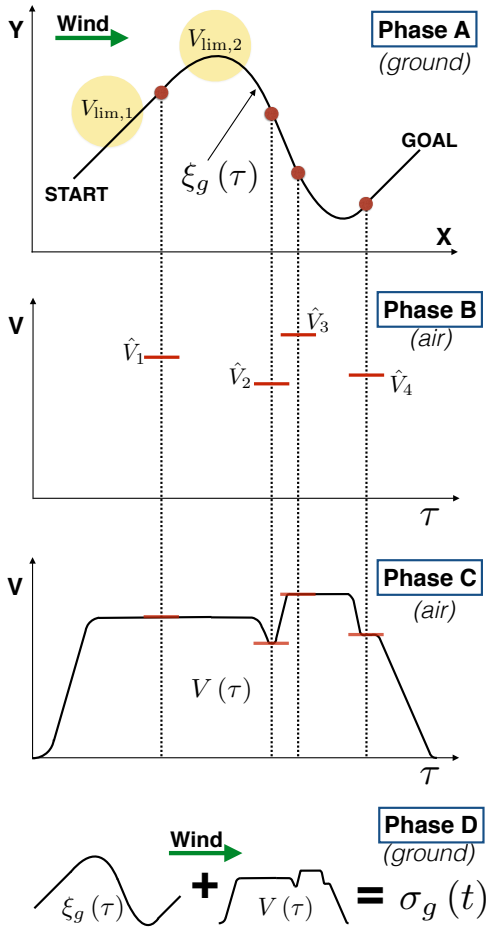


Fig. 3: Overview of κITE

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.

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.

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 end-points.

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 (Ref. 11)):

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

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 $\texttt{HeadingInAir}\left(\psi_g, v, v_w\right)$:

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

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

Arc Parameterization Let s be the arc-length of a path. Let $\xi(s)$ be an arclength parameterized path. Building on (Ref. 10), 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 $\texttt{CurvPoly}\left(\kappa_0, \kappa_f, S_f, \kappa_{max}, \kappa'_{max}, \kappa''_{max}\right)$:

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

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.

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

```
1  for v ← [v_max, v_min] do
2      (κ_max, κ'_max, κ''_max) ←
         CurvLimit(v, φ_max, φ̇_max, φ̈_max, a_max)
3      Δψ ← HeadingInAir(∠(w_{i+1}, w_{i+2})) −
         HeadingInAir(∠(w_i, w_{i+1}))
4      for κ ← [κ_min, κ_max] do
5          κ_2(s) = κ
6          for S_f^1 ← [S_f^min, S_f^max] do
7              κ_1(s) ← CurvPoly(0, κ, S_f^1, κ_max, κ'_max, κ''_max)
8              if κ_1(s) ∈ ∅ then
9                  break
10         S_f^3 ← S_f^1
11         κ_3(s) ← CurvPoly(κ, 0, S_f^3, κ_max, κ'_max, κ''_max)
12         S_f^2 = (Δψ − ∫_0^{S_f^1} κ_1(s) − ∫_0^{S_f^3} κ_3(s)) / κ
13         if S_f^2 ≥ 0 then
14             break
15     κ(s) ← [κ_1(s)  κ_2(s)  κ_3(s)]
16     ξ^i_arc,gnd(s) ← CurvPolyGnd(κ(s))
17     if Σ_{j=i}^{i+1} 𝕀_j(ξ^i_arc,gnd(s)) > 0 then
18         break
19 return v, ξ^i_arc(s), ξ^i_arc,gnd(s)
```

Phase B: Time Optimization

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

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

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-

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

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

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

Algorithm for Initialization Algorithm 2 is used to feasibly initialize the nonlinear optimization problem above, which results in significant improvements in convergence rates. It uses

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

$\left\{\hat{V}_i\right\} \leftarrow \left\{V_{start}, \left\{V_{pt,i}\right\}, V_{goal}\right\}$; $Visited \leftarrow \{0\}_{N+2}$;
$\hat{V}_1 \leftarrow$ `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 function `MakeFeasible` $\left(\hat{V}_1, \hat{V}_2, a, S\right)$, defined as:

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

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.

RESULTS

Our algorithm is currently part of the C++-based motion planning architecture running on board the full-scale autonomous helicopter with a human pilot-in-the-loop described previously. The heliopter 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 (Ref. 12) 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. Fig. 4 depicts the inputs to the algorithm.

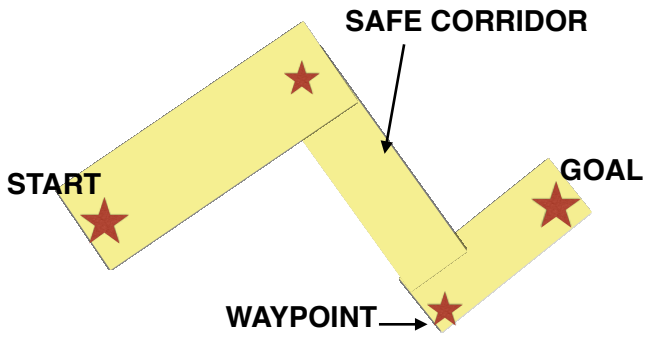


Fig. 4: Overview of the problem, showing waypoints and the safe flight corridor.

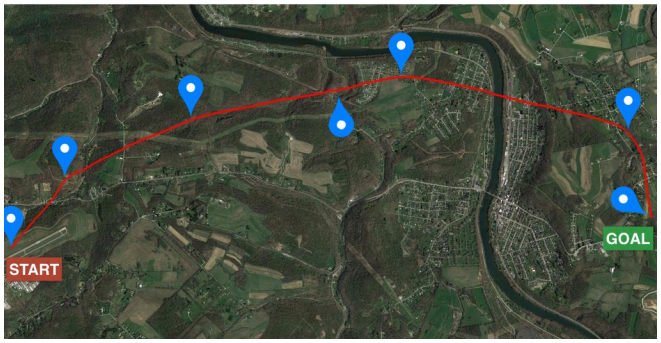
Performance In Wind

Fig. 5 shows the mission waypoints and planned trajectory for a flight test performed with our system in a $\sim$ 20 knots wind blowing towards 270°N. Fig. 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. 7 shows similar speed and heading performance plots for another flight test with a $\sim$ 19 knots wind blowing towards 80°N.

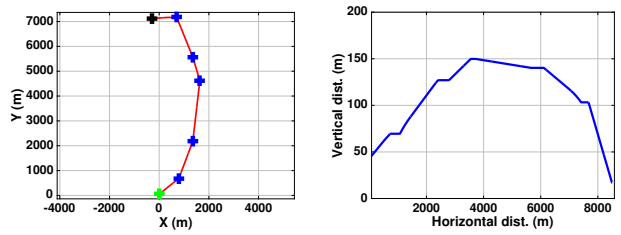
Our system has thus far been tested in 23 flights under winds upto 40 knots, where the algorithm failed to compute a feasible trajectory in 3 cases. The failures were due to very tight corridors, where the algorithm could not keep the trajectory in safe airspace under the corresponding wind conditions.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

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. 8a shows plots of wind direction and magnitude from one of our test flights, and compares the spatial profile of the trajectory (b) 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



(a)



(b)

Fig. 5: Results from a test conducted on our full-size helicopter with a ~ 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.

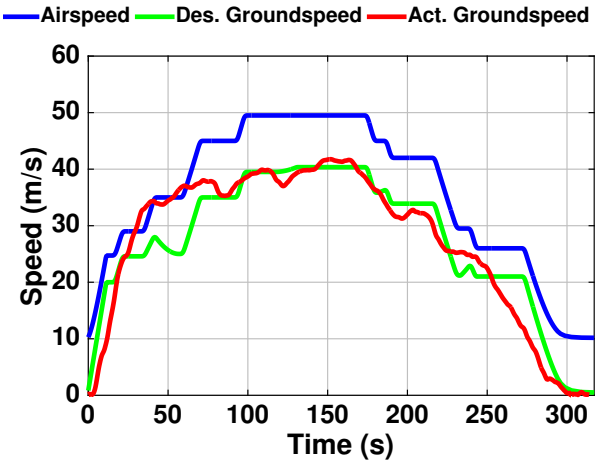
weather station) all along the mission to compute a feasible trajectory, it is essential for it to re-compute its trajectory online when the measured wind deviates significantly from the original estimates. In all our tests, κITE returns a solution (or failure code in the three runs mentioned above) within $5-7s$.

CONCLUSIONS

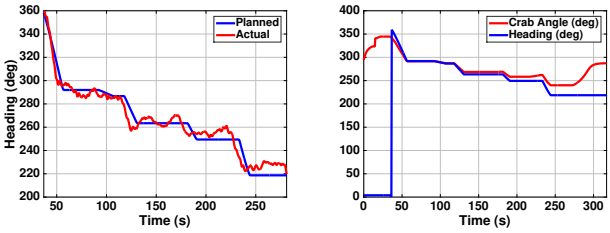
Our conclusions can be summarized as follows-

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.

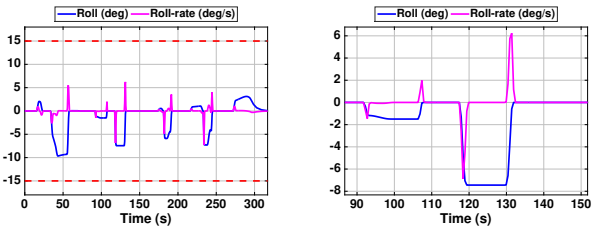
2. Decoupling path optimization from velocity optimization is an extremely effective strategy to make the highly non-convex problem tractable. Our derivation of appropriate curvature limits, taking into account all appropriate dynamic limits of the system (roll and its derivatives, airspeed and acceleration) 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.



(a)



(b)

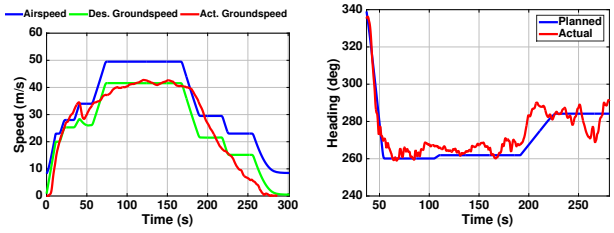


(c)

Fig. 6: Speed, heading and roll profiles for the flight test shown in Fig. 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.

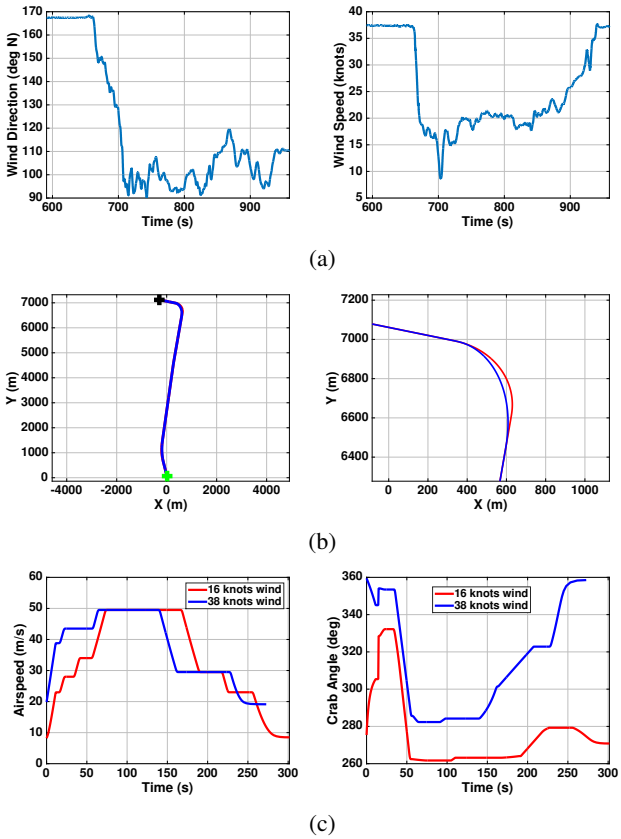
3. We have demonstrated the effectiveness of our algorithm with 20 successful real-world flight tests using both on-board wind measurements and estimates from weather stations, and analyzed the 3 cases where it failed to find a feasible trajectory.

4. 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.



(a)

Fig. 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).



(b)

(c)

Fig. 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.

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. We have open-sourced a MATLAB implementation of κITE at https://bitbucket.org/castacks/kite_optimizer.

Author contact: Vishal Dugar (vdugar@andrew.cmu.edu), Sanjiban Choudhury (sanjibac@andrew.cmu.edu), and Sebastian Scherer (basti@andrew.cmu.edu).

ACKNOWLEDGMENTS

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 43-2249-16.

REFERENCES

[1]Althoff, D., Althoff, M., and Scherer, S., "Online safety verification of trajectories for unmanned flight with offline computed robust invariant sets," Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on, 2015.

[2]Otte, M., Silva, W., and Frew, E., "Any-Time Path-Planning: Time-Varying Wind Field + Moving Obstacles," IEEE International Conference on Robotics and Automation, 2016.

[3]Choudhury, S., Arora, S., and Scherer, S., "The planner ensemble: Motion planning by executing diverse algorithms," 2015 IEEE International Conference on Robotics and Automation (ICRA), 2015.

[4]Yoo, C., Fitch, R., and Sukkarieh, S., "Online task planning and control for fuel-constrained aerial robots in wind fields," *The International Journal of Robotics Research*, 2015, pp. 0278364915595278.

[5]Goerzen, C., Kong, Z., and Mettler, B., "A survey of motion planning algorithms from the perspective of autonomous UAV guidance," *Journal of Intelligent and Robotic Systems*, Vol. 57, (1-4), 2010, pp. 65–100.

[6]Dugar , V., Choudhury, S., and Scherer, S., "A KITE in the Wind: Smooth Trajectory Optimization in a Moving Reference Frame," IEEE International Conference on Robotics and Automation, May 2017.

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

[8]Bobrow, J. E., Dubowsky, S., and Gibson, J., "Time-optimal control of robotic manipulators along specified paths," *The international journal of robotics research*, Vol. 4, (3), 1985, pp. 3–17.

[9]hwan Jeon, J., Cowlagi, R. V., Peters, S. C., Karaman, S., Frazzoli, E., Tsiotras, P., and Iagnemma, K., "Optimal motion planning with the half-car dynamical model for autonomous high-speed driving," 2013 American Control Conference, 2013.

[10]Kelly, A. and Nagy, B., "Reactive nonholonomic trajectory generation via parametric optimal control," *The International Journal of Robotics Research*, Vol. 22, (7-8), 2003, pp. 583–601.

[11]Dugar, V., "Additional proofs," https://goo.gl/AP3OG5.

[12]Choudhury , S., Arora, S., and Scherer, S., "The Planner Ensemble and Trajectory Executive: A High Performance Motion Planning System with Guaranteed Safety," AHS 70th Annual Forum, Montre al, Que bec, Canada, May 2014.

[13]Prouty, R. W., *Helicopter performance, stability, and control*, 1995.

[14]FAA, *Helicopter Flying Handbook*.

[15]Seleck, M., Vana, P., Rollo, M., and Meiser, T., "Wind corrections in flight path planning," *Int J Adv Robotic Sy*, Vol. 10, (248), 2013.

[16]McGee, T. G., Spry, S., and Hedrick, J. K., "Optimal path planning in a constant wind with a bounded turning rate," AIAA Guidance, Navigation, and Control Conference and Exhibit, 2005.

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

[18]Bakolas, E. and Tsiotras, P., "Time-optimal synthesis for the Zermelo–Markov–Dubins problem: the constant wind case," American Control Conference (ACC), 2010.

[19]Bakolas, E. and Tsiotras, P., "Optimal synthesis of the Zermelo–Markov–Dubins problem in a constant drift field," *Journal of Optimization Theory and Applications*, Vol. 156, (2), 2013, pp. 469–492.

[20]Anderson, E. P., Beard, R. W., and McLain, T. W., "Real-time dynamic trajectory smoothing for unmanned air vehicles," *IEEE Trans. Contr. Sys. Techn.*, Vol. 13, 2005, pp. 471–477.

[21]Jung, D. and Tsiotras, P., "On-line path generation for small unmanned aerial vehicles using B-spline path templates," AIAA Guidance, Navigation and Control Conference, AIAA, Vol. 7135, 2008.

[22]Frazzoli, E., Dahleh, M. A., and Feron, E., "Real-time motion planning for agile autonomous vehicles," *Journal of Guidance, Control, and Dynamics*, Vol. 25, (1), 2002, pp. 116–129.

[23]Karaman, S. and Frazzoli, E., "Incremental sampling-based algorithms for optimal motion planning," *Robotics Science and Systems VI*, Vol. 104, 2010.

[24]Techy, L., Woolsey, C. A., and Morgansen, K. A., "Planar path planning for flight vehicles in wind with turn rate and acceleration bounds," Robotics and Automation (ICRA), 2010 IEEE International Conference on, 2010.

[25]Lipp, T. and Boyd, S., "Minimum-time speed optimisation over a fixed path," *International Journal of Control*, Vol. 87, (6), 2014, pp. 1297–1311.

[26]Verscheure, D., Demeulenaere, B., Swevers, J., De Schutter, J., and Diehl, M., "Time-optimal path tracking for robots: A convex optimization approach," *IEEE Transactions on Automatic Control*, Vol. 54, (10), 2009, pp. 2318–2327.