Special Issue: Dynamic Large-Scale Swarm Systems in Urban Environments: Results from the DARPA OFFSET Program

Regular Article

Agile Fixed-Wing UAVs for Urban Swarm Operations

Max Basescu, Adam Polevoy, Bryanna Yeh, Luca Scheuer, Erin Sutton and Joseph L. Moore

Johns Hopkins University Applied Physics Lab, Laurel, Maryland 20723, USA

Abstract: Fixed-wing unmanned aerial vehicles (UAVs) offer significant performance advantages over rotary-wing UAVs in terms of speed, endurance, and efficiency. Such attributes make these vehicles ideally suited for long-range or high-speed reconnaissance operations and position them as valuable complementary members of a heterogeneous multi-robot team. However, these vehicles have traditionally been severely limited with regards to both vertical take-off and landing (VTOL) as well as maneuverability, which greatly restricts their utility in environments characterized by complex obstacle fields (e.g., forests or urban centers).

This paper describes a set of algorithms and hardware advancements that enable agile fixed-wing UAVs to operate as members of a swarm in complex urban environments. At the core of our approach is a direct nonlinear model predictive control (NMPC) algorithm that is capable of controlling fixed-wing UAVs through aggressive post-stall maneuvers. We demonstrate in hardware how our online planning and control technique can enable navigation through tight corridors and in close proximity to obstacles. We also demonstrate how our approach can be combined with onboard stereo vision to enable high-speed flight in unknown environments. Finally, we describe our method for achieving swarm system integration; this includes a gimballed propeller design to facilitate automatic take-off, a precision deep-stall landing capability, multi-vehicle collision avoidance, and software integration with an existing swarm architecture.

Keywords: aerial robotics, obstacle avoidance, navigation, robot teaming

1. Introduction

Autonomous robot swarms have the potential to transform many domains where robotic systems already play an important role, including search and rescue, disaster response, transportation, and defense. While research into multi-robot and swarm control has been conducted for many years (Murray, 2007; Keller et al., 2016), few efforts have explored the impact of high-speed, highly dynamic robotic systems on swarm operations in complex real world environments. Rather, UAV swarms have typically consisted of multi-rotors maneuvering at modest speeds in obstacle fields

Received: 18 June 2022; revised: 21 November 2023; accepted: 7 March 2023; published: 12 May 2023.

Correspondence: Joseph L. Moore, Johns Hopkins University Applied Physics Lab, Laurel, MD 20723,
Email: Joseph.Moore@jhuapl.edu

DOI: https://doi.org/10.55417/fr.2023023

(Zhou et al., 2021) or fixed-wing UAVs operating at high altitudes in open sky with large stand-off distances (Chung et al., 2016).

This dichotomy exists primarily because fixed-wing UAVs are viewed as less maneuverable than multi-rotor systems. In traditional low angle-of-attack flight regimes, fixed wings have a large minimum airspeed and a relatively large minimum turning radius when compared with multi-rotors. This dramatically limits their ability to navigate in complex obstacle fields. Moreover, fixed-wing UAVs often require large areas for launch and recovery due to their limited take-off and landing capabilities, thus greatly complicating swarm deployment logistics.

However, fixed-wing UAVs provide a speed and endurance advantage over their rotary-wing counterparts. For this reason, there are many scenarios where highly agile fixed-wing UAVs capable of navigating through complex environments would be a significant asset to an autonomous swarm. For instance, agile fixed-wing UAVs could dramatically increase the overall tactical range of a swarm system; they have the ability to travel longer distances while also being able to complete many tasks traditionally reserved for multi-rotors (e.g., a low-altitude reconnoiter of a narrow alleyway). These vehicles could also serve as high-speed assets that can out-maneuver adversaries and conduct rapid reconnaissance. For an approximate analysis of agile fixed-wing energy advantage over quadrotors, see Appendix B.

Unlocking the potential of fixed-wing UAVs requires developing planning and control strategies that can exploit the vehicle's full flight envelope. For multi-rotors, aggressive maneuvering can be achieved by leveraging existing differentially-flat dynamics models to plan trajectories in real-time (Mellinger and Kumar, 2011). Such models do not exist for fixed-wing UAVs operating across the full flight envelope. Hybrid vehicle designs (Rehan et al., 2022) are able to improve VTOL capabilities at the expense of complexity and cost but also suffer from the same control challenges as traditional fixed-wing UAVs in dynamic regimes.

In this paper, we present a software and hardware autonomy stack called ACCIPITER, or Aerobatic Control and Collaboration for Improved Performance in Tactical Evasion and Reconnaissance, which seeks to address the major challenges associated with operating agile fixed-wing UAVs as part of an autonomous swarm in complex urban environments. We first describe our underlying nonlinear model predictive (NMPC) approach for exploiting post-stall aerodynamics to enable fixed-wing navigation in urban environments. Our NMPC approach uses direct trajectory optimization (i.e., direct NMPC) to compute dynamically feasible, collision-free trajectories in real-time assuming the availability of full state and map information. We rely on a minimalistic dynamics model and solve a very fast optimization problem while enforcing both state and actuator constraints. We then describe an extension of our NMPC algorithm that uses onboard stereo sensing to enable navigation without an *a priori* map. Next, we show that our vehicle is capable of useful visual feature detection at high speeds and aggressive attitudes. We also describe our solutions to the autonomous deployment and retrieval challenges; this includes a near-vertical take-off capability that leverages a gimballed propeller design and an approach for post-stall landing. Finally, we describe our approach for multi-vehicle collision avoidance and the integration of our vehicle with the CCAST swarm system (Clark et al., 2021). All of our approaches are accompanied by hardware experiments on aerobatic fixed-wing platforms.

Our paper is organized as follows. In Section 2, we provide a summary of related work. In Section 3, we present the details of our technical approach, including both the dynamics model and the formulation of our control algorithm. We analyze the performance of this algorithm in Section 4. Section 5 evaluates performance via hardware experiments in an indoor environment using motion capture. Section 6 tests our approach in outdoor environments with onboard processing, and Section 7 demonstrates flight using onboard stereo vision for collision-free navigation. Then, in Section 8, we discuss the application of our NMPC algorithms to address challenges with urban swarm operations, including take-off, landing, collision avoidance, and swarm system integration. Finally, in Section 9, we discuss our results and propose directions for future work.

We note that in (Basescu and Moore, 2020), we previously presented the core approach and initial validation in Sections 3, 4, and 5. Similarly, in (Polevoy et al., 2022), we presented the vision-based

collision-free navigation approach discussed in Section 7. In this journal paper, we expand on this prior research by addressing the challenges associated with integration into an urban swarm system system and presenting additional hardware experiments.

2. Related Work

Unmanned aerial vehicles have been the subject of motion planning and control research for several decades. This research has been motivated in part by the desire to leverage UAVs as a mobile robotics platform capable of navigating in three dimensions. Because of their mechanical simplicity, ease of use, and vertical take-off and landing (VTOL) capabilities, multi-rotor UAVs, and in particular quadcopter UAVs, have become the platform of choice for advancing various aspects of mobile robotics. In contrast, while fixed-wing UAVs have been explored as a platform for overhead surveillance, far less work has focused on navigating fixed-wing UAVs in constrained environments. Both systems have been used in the context of aerial swarms, though multi-rotors are often the platform of choice when a swarm must operate in close proximity to obstacles.

2.1. Quadcopter Planning and Control

Due to the prevalence of quadcopter UAVs as a mobile robotics research platform, approaches for quadcopter planning and control have advanced more quickly than other classes of unmanned aerial systems. Many of the approaches to quadcopter control rely on a hierarchical control paradigm. In many cases, an inner control loop is used to regulate attitude, where the assumption is made that attitude regulation occurs at a much higher frequency than velocity regulation. Some approaches [e.g., (Waslander et al., 2005; Hoffmann et al., 2007)] employ linear control, often in the form of PID control, to regulate attitude. Other approaches such as (Altug et al., 2002) employ feedback linearization to control the attitude loop. In these hierarchical controller designs, feedback linearization is almost always employed to regulate translational velocities and position (Hoffmann et al., 2011). In moderately aggressive flight control regimes (attitudes $\leq 30°$), reasonable position tracking, even at high speeds, can be achieved using this hierarchical approach [see (Hoffmann et al., 2011)].

For more aggressive, aerobatic maneuvers, where significant deviation from small angle approximations is required, greater attention must be paid to reasoning about the full dynamics. In (Mellinger and Kumar, 2011), the authors use a differentially flat nonlinear quadcopter dynamics model and derive a "minimum-snap" trajectory optimization process. This approach reduces the trajectory generation problem to computing a set of piecewise polynomial splines with sufficient smoothness. The minimum-snap trajectory approach has become a preferred solution for quadcopter control, resulting in a number of extensions of the differential flatness formulation to address additional aerodynamic effects such as rotor drag (Faessler et al., 2018). Researchers have also combined minimum-snap trajectory design more directly with approaches for more global motion planning. In (Richter et al., 2016), the researchers combine a RRT* approach with polynomial trajectories to achieve a dynamically a feasible motion plan. In (Deits and Tedrake, 2015), researchers use free-space segmentation to plan polynomial trajectories within convex regions.

Researchers have also explored using real-time nonlinear model predictive control with quadrotor UAVs. In (Neunert et al., 2016), the authors develop a model predictive control approach that can plan trajectories in real-time and navigate through narrow windows. In (Falanga et al., 2018), the authors use trajectory optimization to achieve perception-aware control by minimizing a cost-function that encourages information gain.

2.2. Fixed-Wing Planning and Control

Despite their ubiquity in commercial domains, fixed-wing UAVs have received significantly less attention than rotorcraft UAVs in the world of mobile robotics (Liew et al., 2017). Research on

robotic fixed-wing UAVs can generally be divided into two categories: low angle-of-attack flight (Section 2.2.1) and flight across the entire envelope, including post-stall operation (Section 2.2.2).

2.2.1. Low angle-of-attack

Like their quadcopter counterparts, fixed-wing UAVs operating in conventional flight regimes leverage a hierarchical control paradigm, where an inner attitude regulation loop is driven by a an outer control loop that regulates velocity and heading. These methods for trajectory generation and tracking at low angles of attack include the widely used L1 guidance controller (Park et al., 2004) and NMPC with a cascaded control hierarchy (Stastny and Siegwart, 2018).

For more aggressive maneuvers, it is often necessary to avoid a hierarchical control approach. In many cases, researchers leverage a trajectory library approach when controlling fixed-wing UAVs. For instance, in (Barry et al., 2018), a trajectory and trim library was used for aggressive, high-speed navigation around trees. Motion planning for fixed-wing UAVs was also explored in (Majumdar and Tedrake, 2017), where a library of funnels exploited invariance of initial conditions to navigate safely through a three dimensional obstacle field in real-time. However, this library only sparsely covered a relatively narrow region of the state space in order to maintain computational tractability during the online planning stage. While these trajectory libraries can accommodate post-stall flight, they are more effective in low angle-of-attack regimes where trajectories can be sampled from a reduced region of state space.

Researchers have also explored real-time trajectory optimization for fixed-wing UAVs at low angles of attack via differential flatness. Although a general three dimensional aerodynamic model for fixed wings can be relatively complex and computationally expensive for use in real-time control problems (Selig, 2014), strong simplifications can be made under the assumption of low angle-of-attack flight. In (Alturbeh and Whidborne, 2014), researchers exploited differential flatness of a point-mass model to generate B-spline trajectories in real-time. Differential flatness of a similar simplified model for coordinated flight (Hauser and Hindman, 1997) was leveraged in (Bry et al., 2015), in which real-time motion planning used a Dubins-Polynomial trajectory representation. However, in order to guarantee that input constraints on the underlying physical system are respected, the performance envelope of these methods is limited even beyond the explicit assumptions in the models. This is in stark contrast with similar methods for rotorcraft UAVs which leverage a tighter coupling between the control inputs of the differentially flat system and the physical actuator commands (Mellinger et al., 2012).

2.2.2. Post-stall regime

Some of the earliest and most notable work in post-stall maneuvers with fixed-wing UAVs is presented in (Sobolic and How, 2009), where researchers explored a transition maneuver to enter and exit a prop-hang configuration. This was preceded by research in (Milam et al., 2002), which sought to generate real-time trajectories for a ducted-fan UAV with a planarized aerodynamic model. High angle-of-attack perching maneuvers were executed using an optimal policy obtained from value iteration in (Cory and Tedrake, 2008). Further work on the perching problem was conducted in (Moore et al., 2014), where a library of trajectories generated offline was used to enable robust perching performance.

Other types of maneuvers for fixed-wing UAVs such as aggressive turn-arounds have been examined in (Matsumoto et al., 2010; Levin et al., 2016; Levin et al., 2019). A high fidelity physics model suitable for control of fixed-wing UAVs through the entire flight envelope was proposed in (Khan and Nahon, 2016). This model uses a component breakdown approach, in which the forces and moments produced by individual segments of each aerodynamic surface are computed and then superimposed. Unsteady aerodynamic effects as well as motor and propeller physics are also incorporated. This model has been used to generate and follow knife-edge trajectories in (Levin et al., 2017), and to track aggressive maneuvers in (Bulka and Nahon, 2018) and (Hernández Ramírez and Nahon, 2020). Several recent publications (Levin et al., 2019; Bulka and Nahon, 2019) have also leveraged this model for real-time motion planners using libraries consisting of trim primitives and/or

trajectories generated offline. As with (Barry et al., 2018) and (Majumdar and Tedrake, 2017), these libraries are relatively coarse in their coverage of the state space.

In (Park, 2020) the author uses an inner-loop attitude controller and an outer-loop a virtual-target-following guidance method to achieve precision deep-stall landings. The authors of (Mathisen et al., 2016) performed a non-real-time simulation analysis where input and state trajectories for a full 6-DOF aircraft model were optimized using NMPC to achieve precision deep-stall landings with slow touch-down speeds. Further efforts in (Mathisen et al., 2020) examined NMPC for a planarized model, however they noted that even by computing only a single iteration of the nonlinear optimization problem per control step, their approach is still up to five times too slow to run in real-time.

2.3. UAV Navigation with Onboard Sensors

A majority of the research into UAV navigation and control using onboard sensing has been conducted with multi-rotor UAVs. Early research in this area focused on LIDAR-based simultaneous localization and mapping (SLAM), and leveraging generated maps for navigation and collision avoidance (Bachrach et al., 2011; Shen et al., 2011; Grzonka et al., 2009). As the flight speeds increased, onboard vision emerged as another primary sensing solution (Blösch et al., 2010; Faessler et al., 2016; Huang et al., 2017). Researchers also began to address the challenges associated with the computational costs of online mapping, as well as the challenges posed by large amounts of state uncertainty. (Tabib and Michael, 2021) utilizes Gaussian Mixture Model (GMM) based mapping to reduce the memory requirements of storing and transmitting mapping data. (Gao et al., 2019) addresses the mapping challenge by maintaining a KD-tree representation of LIDAR measurements, generating a valid flight corridor, and optimizing trajectories within that corridor. Similarly, (Florence et al., 2020) utilizes a trajectory library approach on a KD-tree representation of current depth camera measurements to minimize the probability of collision. The authors then extend this work with a lightweight mapping data structure, NanoMap (Florence et al., 2018), which maintains a history of sensor measurements.

Several efforts have explored the use of onboard sensing for fixed-wing navigation. In (Bry et al., 2015), the authors demonstrated the ability to fly in a constrained-space using a scanning LIDAR. Their work focused on the state-estimation problem and used a differentially flat model valid for a reduced flight envelope. In (Barry et al., 2018), authors demonstrated fixed-wing flight using onboard stereo vision and a trajectory-library approach limited to conventional angle-of-attack regimes.

2.4. Control of Multiple Dynamic Aerial Vehicles

With a few exceptions [e.g., (Hamer et al., 2018; Turpin et al., 2014)], approaches to swarm control have been surprisingly limited in their ability to reason about the dynamics of individual agents. Typically, swarm research has sought to develop coordinated behaviors for large numbers of agents (Preiss et al., 2017) or to solve the multi-agent collision-free path planning problem (Arul et al., 2019). However, this has often been done with extremely simplified dynamic or even kinematic models (Chung et al., 2018). Even in the context of aerial swarm-on-swarm combat, agents have maintained fairly conservative flight envelopes (McGrew et al., 2010; Strickland et al., 2018).

3. Approach

Our approach is based on the hypothesis that to successfully navigate in complex, dynamic environments which are unknown *a priori*, the fixed-wing UAV must be able to generate and track sophisticated trajectories in real-time that exploit the full flight envelope and achieve "supermaneuverability."[1] For instance, in post-stall flight (i.e., when the airflow is separated from the

[1] Supermaneuverability is defined in (Herbst, 1984) as the ability "to execute tactical maneuvers with controlled side slipping and at angles of attack beyond maximum lift."

(a) **(b)**

Figure 1. (a) The image shows an aerobatic aircraft executing a post-stall 90° turn through a virtual hallway denoted by two striped poles. (b) Plot shows how fixed-wing UAVs achieve smaller turn radii by executing high angle-of-attack maneuvers. These curves were generated by computing turn trim conditions using a nonlinear program and a simplified nonlinear aircraft model without control surfaces. The area with the gradient represents fixed-wing UAVs for a range of scales capable of high angle-of-attack maneuvers. The red area represents quadrotors for a range of scales. The dashed black lines represent fixed-wing UAVs with different wing loadings constrained to low angle-of-attack maneuvers. The high angle-of-attack fixed-wing is able to achieve similar minimum turn radii as the quadcopter and significantly smaller minimum turn radii than the low angle-of-attack vehicles. See Appendix A for details.

aircraft's wing), an aircraft can exploit excess drag to rapidly slow down and execute a tight turn, using its thrust to counteract the loss in lift and contribute to the centripetal acceleration. These post-stall maneuvers have the potential to dramatically reduce vehicle turning radius and allow for a fixed-wing vehicle to operate in constrained spaces (e.g., urban centers) and in dense swarming scenarios. Figure 1(a) shows an example of a "post-stall" turn.

To understand how fixed-wing and quadrotor turning radius varies with velocity, angle-of-attack, and vehicle scale, we use a simplified dynamics model and solve a nonlinear program to compute a turn trim condition for a set of fixed-wing and quadcopter UAVs (see Appendix A for details). At high angles of attack, the fixed wing is capable of low turn radii maneuvers on par with the quadcopter UAV [see Figure 1(b)]. In addition, the aerobatic fixed wing can achieve smaller turn-radii than a similarly sized fixed wing restricted to a low angle-of-attack regime. We use the same simplified model to show that this turning radius advantage is not restricted to high thrust-to-weight ratios, but also improves maneuverability for modest thrust-to-weight ratios (see Figure 2). For these analyses, and for the work in this paper, we only consider a "puller" configuration; Such a design feature is important for vehicles to be able to take advantage of the control authority afforded by propeller backwash at low air-speeds. In addition to improving turning radius, post-stall maneuvers are also useful for achieving short landings [e.g., via post-stall perching (Moore et al., 2014)] and near-vertical take-offs [e.g., (Moore et al., 2018)]. Post-stall regimes are additionally useful for slower cruise speeds, such as might be needed when flying with a limited sensor horizon.

To maximize maneuverability (e.g., minimize turning radius), the UAV controller must reason about a complex nonlinear aerodynamics model for which a useful differentially flat system has not been found. While trajectory libraries are commonly used to reduce online computational burdens for systems with nontrivial dynamics, they often become prohibitively expensive to search in real-time and result in suboptimal solutions when the vehicle state space is large (>6) and the environment is complex (Milam et al., 2000). This broad finding has proven accurate by recent research in fixed-wing motion planning which has been restricted to very limited library sizes (as described in Section 2.2.2).

Therefore, instead of building a high-fidelity physics model of the aircraft, we build a minimalistic medium-fidelity physics model sufficient to capture the nonlinear post-stall aerodynamics and amenable to real-time trajectory optimization. Despite the computational complexity (Dekka et al., 2016), we choose to utilize direct trajectory optimization methods because of their improved

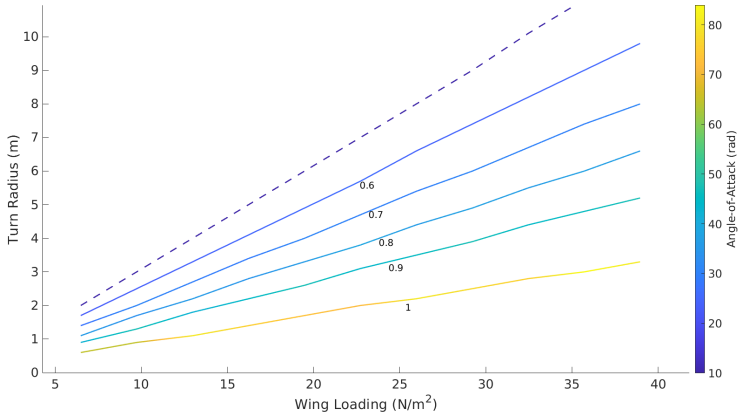


Figure 2. This figure shows how minimum turn radius changes with thrust-to-weight ratio. Up to about 0.6 T/W, post-stall maneuvering still improves the minimum turning radius. The NACA 2412 airfoil stalls between 10° and 15° depending on the Reynolds number. The dashed line represents minimum turning radius assuming attached flow ($\alpha < 10°$). The lines below the dashed line show how minimum turning radius decreases with increasing thrust-to-weight for high angle-of-attack conditions.

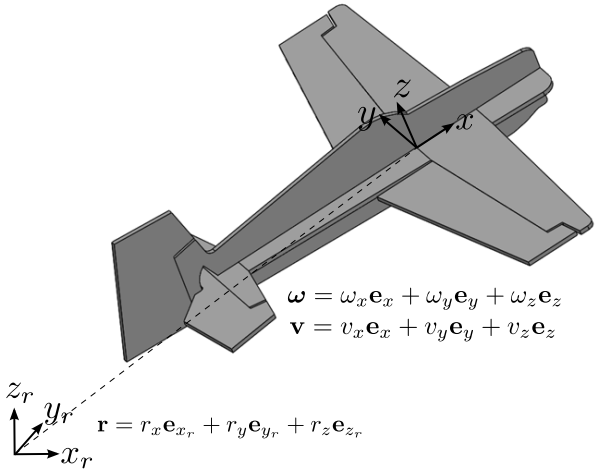


Figure 3. A depiction of the main coordinate frames used for our dynamics model.

numerical conditioning, sparse constraints, and ability of the gradient computations to be trivially parallelized (Tedrake, 2009). Additionally, we preserve the ability to directly apply costs and constraints on the state trajectory. As discussed in (Levin et al., 2017), these direct methods can also be easily seeded by a simplified randomized motion planner. We formulate a nonlinear optimization problem which can be solved at real-time rates and use local linear feedback as an ancillary controller to stabilize the vehicle between replans. We also optimize and control over the entire vehicle state-space, so as to not make assumptions about timescale separations. We refer to our online trajectory optimization approach as direct nonlinear model predictive control, or direct NMPC, following the definition of NMPC in (Johansen, 2011). However, in contrast to typical NMPC formulations, we optimize our trajectory over several sampling intervals.

3.1. Dynamics Model

We extend the planar post-stall aircraft model presented in (Moore et al., 2014) into a three-dimensional model of a fixed-wing UAV with a propeller in a "puller" configuration (see Figure 3).

We define our state as

$$\mathbf{x} = \begin{bmatrix} r_x, r_y, r_z, \phi, \theta, \psi, \delta_{ar}, \delta_e, \delta_r, \delta_t, v_x, v_y, v_z, \omega_x, \omega_y, \omega_z \end{bmatrix}^T. \tag{1}$$

Here $\mathbf{r} = \begin{bmatrix} r_x, r_y, r_z \end{bmatrix}^T$ represents the position of the center of mass in the world frame $O_{x_r y_r z_r}$, $\boldsymbol{\theta} = \begin{bmatrix} \phi, \theta, \psi \end{bmatrix}^T$ represents the set of $x-y-z$ euler angles applied in a ZYX intrinsic rotation sequence, and $\boldsymbol{\delta} = \begin{bmatrix} \delta_{ar}, \delta_e, \delta_r \end{bmatrix}^T$ are control surface deflections for the right aileron, elevator, and rudder respectively. The left aileron deflection is treated as a dependent parameter, with $\delta_{al} = -\delta_{ar}$. δ_t is the magnitude of thrust from the propeller, $\mathbf{v} = \begin{bmatrix} v_x, v_y, v_z \end{bmatrix}^T$ is the velocity of the center of mass in the world frame, and $\boldsymbol{\omega} = \begin{bmatrix} \omega_x, \omega_y, \omega_z \end{bmatrix}^T$ is the angular velocity of the body in the body-fixed frame O_{xyz}. We can then write $\mathbf{x} = \begin{bmatrix} \mathbf{r}, \boldsymbol{\theta}, \boldsymbol{\delta}, \delta_t, \mathbf{v}, \boldsymbol{\omega} \end{bmatrix}^T$, $\mathbf{u}_{cs} = \begin{bmatrix} \omega_{ar}, \omega_e, \omega_r \end{bmatrix}^T$. The equations of motion then become

$$\dot{\mathbf{r}} = \mathbf{v}, \tag{2}$$

$$\dot{\boldsymbol{\theta}} = \mathbf{T}_{\omega}^{-1} \boldsymbol{\omega}, \tag{3}$$

$$\dot{\boldsymbol{\delta}} = \mathbf{u}_{cs}, \tag{4}$$

$$\dot{\delta}_t = a_t \delta_t + b_t u_t, \tag{5}$$

$$\dot{\mathbf{v}} = \mathbf{R}_b^r \mathbf{f} / m, \tag{6}$$

$$\dot{\boldsymbol{\omega}} = \mathbf{J}^{-1} (\mathbf{m} - \boldsymbol{\omega} \times \mathbf{J}\boldsymbol{\omega}). \tag{7}$$

Here m is vehicle mass, $\mathbf{J}$ is the vehicle's inertia tensor with respect to the center of mass, $\mathbf{f}$ are the total forces applied to the vehicle in body-fixed coordinates, $\mathbf{m}$ are the moments applied about the vehicle's center of mass in body-fixed coordinates, and $\mathbf{u}_{cs}$ are the angular velocity inputs of the control surfaces. $\mathbf{R}_b^r$ denotes the rotation of the body-fixed frame with respect to the world frame, and $\mathbf{T}_{\omega}$ is the transformation which maps the euler angle rates to an angular velocity in body-fixed frame. Additionally, we define the velocity in the body-fixed frame $\mathbf{v}_b = \mathbf{R}_b^{r\,T} \mathbf{v}$ for notational simplicity. The forces acting on the vehicle can be written in the body-fixed frame as

$$\mathbf{f} = \sum_i [\mathbf{R}_{s_i}^b \mathbf{f}_{s_i}] - mg \mathbf{R}_b^{r\,T} \mathbf{e}_z + \mathbf{R}_t^b \mathbf{f}_t + \mathbf{f}_d, \tag{8}$$

where $\mathbf{f}_{s_i}$ represents the force due to each aerodynamic surface in the surface frame $O_{x_s y_s z_s}$ and $\mathbf{f}_t$ represents the force due to the propeller and is given as $\mathbf{f}_t = \begin{bmatrix} \delta_t & 0 & 0 \end{bmatrix}^T$. g is the acceleration due to gravity, and $\mathbf{f}_d$ is a drag force which compensates for additional observed profile drag. $\mathbf{R}_t^b$ is the rotation matrix that defines the orientation of the thrust source with respect to the body-fixed frame. $\mathbf{R}_{s_i}^b$ is the rotation matrix that defines the aerodynamic surface reference frame (in which the z axis is the surface normal) with respect to the body-fixed frame. The aerodynamic surfaces used for this model are the wing, the horizontal and vertical fuselage, the horizontal and vertical tail, and the control surfaces.

To model forces on the aerodynamic surfaces, we use

$$\mathbf{f}_{s_i} = f_{n,s_i} \mathbf{e}_z = \frac{1}{2} C_{n_i} \rho |\mathbf{v}_{s_i}|^2 S_i \mathbf{e}_z, \tag{9}$$

where ρ is the density of air, S_i is the surface area, and $\mathbf{v}_{s_i}$ is the velocity of the ith aerodynamic surface in the surface frame given as

$$\mathbf{v}_{s_i} = \mathbf{R}_{s_i}^{b\,T} (\mathbf{v}_b + \boldsymbol{\omega} \times \mathbf{r}_{h_i} + \gamma_i \mathbf{v}_{\text{bw}}) + (\mathbf{R}_{s_i}^{b\,T} \boldsymbol{\omega} + \boldsymbol{\omega}_{s_i}) \times \mathbf{r}_{s_i}. \tag{10}$$

Here $\mathbf{r}_{h_i}$ represents the displacement from the vehicle center of mass to a point on the aerodynamic surface that is stationary in the body-fixed frame. For control surfaces, this point is located at the center of the axis of rotation between the aircraft body and the control surface. For static aerodynamic surfaces, this point is located at the surface centroid. $\mathbf{r}_{s_i}$ represents the displacement from the hinge point to the surface center of pressure in the surface frame, and $\boldsymbol{\omega}_{s_i}$ is the simple

rotation rate of the aerodynamic surface in the aerodynamic surface frame. This is only nonzero for actuated surfaces. $\mathbf{v}_{\text{bw}}$ is the velocity due to the backwash of the propeller. It is approximated using actuator disk theory as

$$\mathbf{v}_{\text{bw}} = \left[\sqrt{\|\mathbf{v}_p\|_2^2 + \frac{2\delta_t}{\rho S_{\text{disk}}}} - \|\mathbf{v}_p\|_2\right]\mathbf{e}_x, \tag{11}$$

where S_{disk} is the area of the actuator disk and $\mathbf{v}_p$ is the freestream velocity at the propeller. γ is an empirically determined backwash velocity coefficient. C_{n_i} comes from the flat plate model in (Hoerner and Borst, 1985) and is given as

$$C_{n_i} = 2\sin\alpha_{s_i} \tag{12}$$

$$\alpha_{s_i} = \arctan\frac{v_{s_i,z}}{v_{s_i,x}}, \tag{13}$$

where α_{s_i} is the angle-of-attack for the i^{th} aerodynamic surface. $\mathbf{m}$ in the body-fixed frame can be given as

$$\mathbf{m} = \sum_i (\mathbf{l}_{s_i} \times \mathbf{R}_{s_i}^b \mathbf{f}_{s_i}), \tag{14}$$

where $\mathbf{l}_{s_i} = \mathbf{r}_{h_i} + \mathbf{R}_{s_i}^b \mathbf{r}_{s_i}$ is the vector from the vehicle center of mass to the surface center of pressure in the body-fixed frame. To model the thrust dynamics, we assume a first order linear model for the thrust as given in (Equation 5) where a_t and b_t are constants and u_t is the normalized control signal for the motor, $u_t \in [0, 1]$.

To account for unmodeled profile drag, we add the drag term

$$\mathbf{f}_d = -\frac{1}{2}C_{b_d}\rho|\mathbf{v}_b|^2\frac{\mathbf{v}_b}{|\mathbf{v}_b|}, \tag{15}$$

where C_{b_d} was determined empirically.

3.2. Control Strategy

To achieve real-time planning for fixed-wing vehicles across the entire flight envelope, we utilize a four-stage, hierarchical control strategy consisting of a randomized motion planner, a spline-based smoothing algorithm, direct trajectory optimization, and local linear feedback control. The spline-based smoothed path is used as a seed to the trajectory optimizer and also provides a receding horizon goal point. The local linear feedback controller compensates for model-uncertainty and environmental disturbances between plans. A diagram of our receding-horizon control approach can be found in Figure 4.

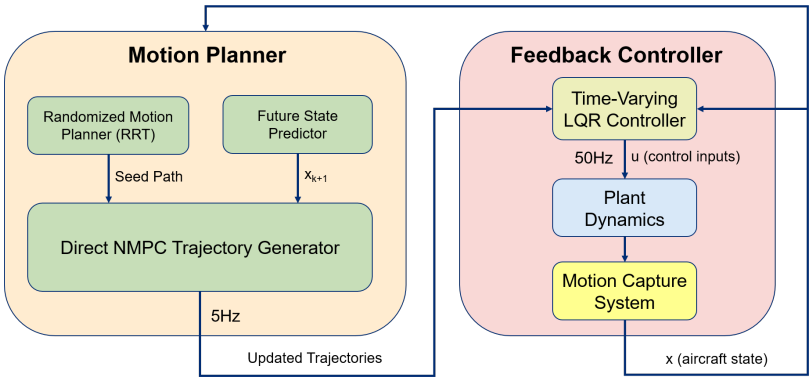


Figure 4. A block diagram of our receding-horizon control approach.

3.2.1. RRT generation and spline-based smoothing

Randomized motion planning has proved to be an effective strategy for generating motion plans in complex environments and in the presence of local minima. Rapidly-exploring Random Trees (RRTs) have been a particularly powerful tool for solving path planning problems. However, RRTs often exhibit poor performance when required to reason about dynamical systems with large state-spaces. Here, we use a simplified three-state model to plan a collision-free "flight path" with a rapidly exploring random tree in three dimensions.

Starting with an arbitrary 3D mesh of an environment, binvox (Min, 2019; Nooruddin and Turk, 2003) is used to perform voxelization. The voxelized mesh is then converted to a binary occupancy tree, and OctoMap (Hornung et al., 2013) is used to provide information about proximity to obstacles. While the trajectory generated by this process will not be a feasible path for our aircraft to follow, it attempts to provide a receding-horizon goal point for the next stage of the planner and helps the trajectory optimizer avoid local minima. Initially, we study the motion planning problem apart from the perception problem and assume that we have a full map of the environment. In Section 7, we extend our approach to the case where the field-of-view is limited and trajectories must be replanned when provided with new information about the environment.

The flight path is generated through a combination of standard RRT (Lavalle, 1998), spline-based smoothing satisfying a maximum curvature constraint (Yang and Sukkarieh, 2010), and a final reparameterization of the trajectory. This combination of steps provides a path which is more dynamically feasible than a raw RRT and also helps to intelligently select an goal point for the trajectory optimizer.

First, the RRT is grown in a three dimensional state space with a 10% bias towards the goal point, until it arrives within an error ball around the goal. The resultant path $\mathbf{x} = [x_0, x_1, \ldots, x_n]$ from start to goal is then pruned as described in (Yang and Sukkarieh, 2008) to remove extraneous nodes.

We then employ G2 Continuous Cubic Bézier Spiral Path Smoothing (G2CBS) to obtain a continuous curvature, smooth path (Yang and Sukkarieh, 2010). This algorithm examines each set of three adjacent nodes $[W_1, W_2, W_3]$ in the pruned path. It transforms them into a local 2D space, computes control points for two symmetric G2 splines satisfying a maximum curvature constraint $\kappa_{\max}$ (set to 2 m^{-1}), then transforms the control points back into 3D. The output of this algorithm is a set of 3D control points for each spline,

$$\begin{aligned}\mathbf{B_c} &= \begin{bmatrix}\mathbf{B_0} & \mathbf{B_1} & \mathbf{B_2} & \mathbf{B_3}\end{bmatrix} \in \mathbb{R}^{3x4} \\ \mathbf{E_c} &= \begin{bmatrix}\mathbf{E_3} & \mathbf{E_2} & \mathbf{E_1} & \mathbf{E_0}\end{bmatrix} \in \mathbb{R}^{3x4}.\end{aligned} \tag{16}$$

A 3rd order Bézier curve can be evaluated by

$$\begin{aligned}\mathbf{F}(s) = (1-s)^3\mathbf{P_0} + 3(1-s)^2 s\mathbf{P_1} \ldots \\ + 3(1-s)s^2\mathbf{P_2} + s^3\mathbf{P_3},\ s \in [0,1],\end{aligned} \tag{17}$$

where $[\mathbf{P_0}, \ldots, \mathbf{P_3}]$ is the set of control points for the spline. This expression allows us to evaluate the B and E splines represented by the control points $\mathbf{B_c}$ and $\mathbf{E_c}$, and to stitch together the overall G2 continuous path $\mathbf{x}$, which consists of straight line segments and spline segments.

We extend this approach to compute the curvature of the entire path, in order to apply a rough kinematic mapping between curvature and velocity. This mapping allows us to reparameterize the flight path in terms of time. By picking an endpoint for the trajectory optimizer based on a constant time along the seed path, the duration of the optimized trajectory is somewhat regularized. This helps to prevent the solver from wasting computational resources planning too far into the future and allows us to provide a more feasible estimate of final velocity to the solver. We compute the first and second order derivatives of $\mathbf{F}$ with respect to s, $\mathbf{F}'$, and $\mathbf{F}''$. Pulling out the individual elements as

$$\mathbf{F}'(s) = \begin{bmatrix}x' & y' & z'\end{bmatrix},\ \mathbf{F}''(s) = \begin{bmatrix}x'' & y'' & z''\end{bmatrix}, \tag{18}$$

we can express the curvature along $\mathbf{F}(s)$ as

$$\kappa = \frac{\sqrt{(z''y' - y''z')^2 + (x''z' - z''x')^2 + (y''x' - x''y')^2}}{(x'^2 + y'^2 + z'^2)^{\frac{3}{2}}} \tag{19}$$

with the curvature along the straight segments of path equal to 0. The curvature along the entire path is then mapped to velocity as

$$v(s) = \frac{d\mathbf{x}}{dt}(s) = v_{\max} - \kappa(s) * m, s \in [0, s_p], \tag{20}$$

where $\mathbf{x}(s)$ is the full path as a function of s, $v_{\max}$ is the maximum velocity for the kinematic model, m is the linear kinematic mapping parameter which relates curvature to velocity based on the curve in Figure 1(b), and s_p is the overall path length. We can reparameterize by time with

$$t = \int_0^s \frac{1}{v(s)} ds, s \in [0, s_p], \tag{21}$$

which yields a numerically solvable relationship between t and s.

To select an endpoint to feed into the trajectory optimization step, we simply evaluate the position and velocity equations at the time horizon T_H.

3.2.2. Direct trajectory optimization

Direct trajectory optimization formulates the nonlinear optimization problem by including both inputs and states as decision variables (Tedrake, 2009). A collocation or transcription of the dynamics is then used to constrain the variables to ensure dynamic feasibility. For this reason, direct trajectory optimizers are well suited to accept a set of state variable initializations and are are often more robust to local minima than indirect methods (Tedrake, 2009). Here, we use a direct transcription formulation of Simpson's integration rule as described in (Pardo et al., 2016) and formulate a feasibility problem—our objective is to ensure dynamic feasibility of the trajectory and collision avoidance. To solve our direct trajectory feasibility problem, we employ the Sparse Nonlinear Optimizer (SNOPT) (Gill et al., 2005). Collision avoidance is formulated as a nonpenetration constraint on the occupancy grid. In general, we have found these implicit integration constraints to require far fewer knot points, provide good numerical stability, and offer improved performance over spline-based collocation constraints.

Our feasibility problem can be written as

$$\begin{aligned}
\min_{\mathbf{x}_k, \mathbf{u}_k, h} \quad & 0 \\
\text{s.t.} \quad & \forall k \in [0, \ldots, N] \text{ and} \\
& \mathbf{x}_k - \mathbf{x}_{k+1} + \frac{h}{6.0}(\dot{\mathbf{x}}_k + 4\dot{\mathbf{x}}_{c,k} + \dot{\mathbf{x}}_{k+1}) = 0, \\
& \mathbf{x}_f - \boldsymbol{\delta}_f \leq \mathbf{x}_N \leq \mathbf{x}_f + \boldsymbol{\delta}_f, \\
& \mathbf{x}_i - \boldsymbol{\delta}_i \leq \mathbf{x}_0 \leq \mathbf{x}_i + \boldsymbol{\delta}_i, \\
& \mathbf{x}_{\min} \leq \mathbf{x}_k \leq \mathbf{x}_{\max}, \quad \mathbf{u}_{\min} \leq \mathbf{u}_k \leq \mathbf{u}_{\max}, \\
& d(\mathbf{x}) \geq r, \\
& h_{\min} \leq h \leq h_{\max},
\end{aligned} \tag{22}$$

where

$$\begin{aligned}
\dot{\mathbf{x}}_k &= \mathbf{f}(t, \mathbf{x}_k, \mathbf{u}_k), \quad \dot{\mathbf{x}}_{k+1} = \mathbf{f}(t, \mathbf{x}_{k+1}, \mathbf{u}_{k+1}), \\
\mathbf{u}_{c,k} &= (\mathbf{u}_k + \mathbf{u}_{k+1})/2, \\
\mathbf{x}_{c,k} &= (\mathbf{x}_k + \mathbf{x}_{k+1})/2 + h(\dot{\mathbf{x}}_k - \dot{\mathbf{x}}_{k+1})/8, \\
\dot{\mathbf{x}}_{c,k} &= \mathbf{f}(t, \mathbf{x}_{c,k}, \mathbf{u}_{c,k}).
\end{aligned} \tag{23}$$

Here, $\mathbf{x}$ signifies the system state and $\mathbf{u}$ is the vector of control actions. h is the time step bounded from $h_{\min} = 0.001s$ to $h_{\max} = 0.2s$. $\boldsymbol{\delta}_f$ and $\boldsymbol{\delta}_i$ represent the bounds on the desired final and initial states ($\mathbf{x}_f$,$\mathbf{x}_i$), respectively. N is the number of knot points. The minimum distance function d is generated using a distance map computed from a map such as an OctoMap (Hornung et al., 2013).

3.2.3. Local linear feedback control

To control the aircraft to a trajectory between plans, we use time-varying LQR (TVLQR). The control action is given as

$$\mathbf{u}(t, \mathbf{x}) = \mathbf{K}(\mathbf{x} - \mathbf{x}_0(t)) + \mathbf{u}_0(t). \tag{24}$$

$\mathbf{K}(t)$ is the time-dependent feedback gain matrix and is found by integrating

$$-\dot{\mathbf{S}}(t) = \mathbf{A}(t)^T\mathbf{S}(t) + \mathbf{S}(t)\mathbf{A}(t) \ldots \\ -\mathbf{S}(t)\mathbf{B}(t)\mathbf{R}^{-1}\mathbf{B}(t)^T\mathbf{S}(t) + \mathbf{Q} \tag{25}$$

backwards in time from $t = T$ to $t = 0$. Here $\mathbf{A}(t) = \frac{\partial \mathbf{f}(\mathbf{x}_0, \mathbf{u}_0(t))}{\partial \mathbf{x}}$, $\mathbf{B}(t) = \frac{\partial \mathbf{f}(\mathbf{x}_0(t), \mathbf{u}_0(t))}{\partial \mathbf{u}}$, and

$$\mathbf{K}(t) = \mathbf{R}^{-1}\mathbf{B}(t)^T\mathbf{S}(t). \tag{26}$$

Here $\mathbf{x}_0(t)$ and $\mathbf{u}_0(t)$ signify the nominal trajectories. $\mathbf{Q}$ and $\mathbf{R}$ are the weighting matrices for state and action respectively. $\mathbf{S}(t)$ is defined by a final cost matrix $\mathbf{Q}_f$.

4. Real-time Trajectory Optimization Performance

To evaluate the performance of our NMPC approach, we explored trajectory optimization for a post-stall 90° turn using our fixed-wing UAV model. We considered this 90° turn to be a representative maneuver for aerobatic post-stall motion planning. We considered two metrics—computation speed and trajectory following performance. Direct transcription methods, while computationally efficient (i.e., able to exploit sparsity), do not provide the same kind of accuracy as is provided by some shooting and direct pseudospectral methods. Therefore, we did not use integration error as a metric, but rather error in following a trajectory using local feedback. LQR provided this metric as a cost-to-go, and to evaluate trajectory following, we examined the final cost of a simulated vehicle following this trajectory using TVLQR. With regards to computation time, we looked at the time required to find an optimal solution when a naive seed was provided (i.e., linear progression of states from initial state to the goal) and when SNOPT's warm-start feature was used with a feasible seed. To test the warm-start, we sampled from a set of random-initial condition states and examined the time-required to reoptimize the trajectory. We compared a simple Euler integration constraint [e.g., $\mathbf{x}_{k+1} - \mathbf{x}_k - h(\mathbf{x_k}, \mathbf{u_k}) = \mathbf{0}$] with the Hermite-Simpson integration constraint. We did not explore higher order implicit integration constraints due to the associated computational cost.

As can be observed in Figure 5, the direct method using the Euler integration constraint did not become feasible until ≈20 knot points were used. In general, the Hermite-Simpson integration constraint outperformed the Euler integration constraint in computation time by a factor of almost 2-1 for both the naive seed optimization and warm-start optimization, especially below 30 knot points. The Hermite-Simpson approach showed good trajectory following at 10 knot points, while the Euler integration method did not show comparable trajectory following until 24 knot points. SNOPT's warm-start also showed a 4:1 improvement over the naive seeding.

Given these results, we chose to use Hermite-Simpson integration as our integration method with 10 knot points. This provided the best computational performance (≈0.1s for a warm start) and reasonably good trajectory following.

5. Motion Capture Experiments

Given the fidelity limitations of the dynamics model used for control, real-world experiments were imperative in validating the efficacy of the controller design. We first conducted a series

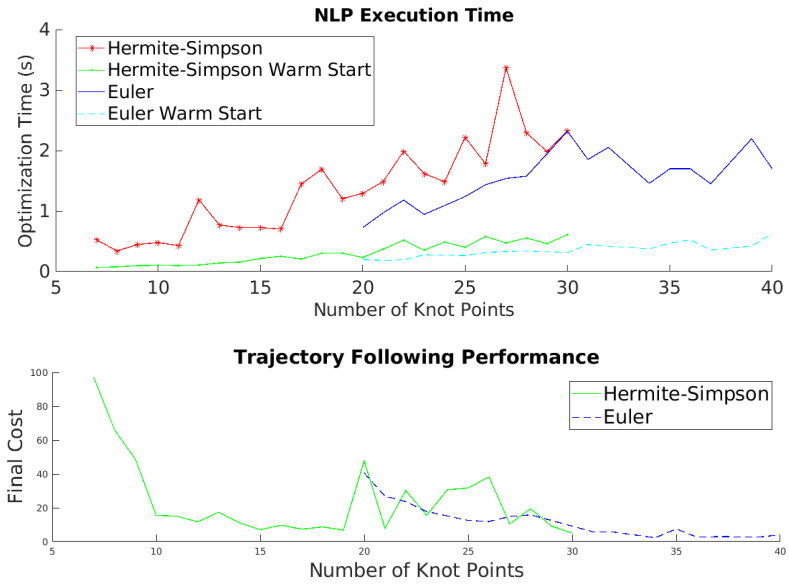


Figure 5. (Above) Plot shows time to compute an optimal trajectory using Hermite-Simpson (red star, green dot) and Euler integration methods (blue line, light blue dash). There is a significant difference between the time needed to compute a trajectory using a naive seed and using a warm-start. The trajectories generated using Euler integration constraints only become feasible at 20 knot points. (Below) Final cost for trajectory following of trajectories generated using Hermite-Simpson (blue) and Euler (green) integration.

Figure 6. 24" wingspan Edge 540 EPP model.

of experiments using a small, 24-inch wingspan foam aircraft (Figure 6) operating in an indoor motion-capture space. For these experiments, all state estimation and control computation was performed offboard the vehicle in real-time.

5.1. Experimental Setup

The plane used for the experiments was a 24-inch wingspan aerobatic Edge 540 model produced by Twisted Hobbys (Figure 6), weighing 120 g. It has an independently controllable rudder and elevator, as well as a pair of kinematically linked ailerons. A 13 g 2300 Kv Crack Series out-runner motor was paired with a 7x3.5 GWS propeller for the thruster. In addition to the stock components for the aircraft, five Vicon markers were attached at various locations on the airframe. A custom bungee

launcher was built to produce consistent initial conditions for each trial, and an adapter designed to interface with the launcher was attached to the underside of the plane. Maximum throws were measured for each control surface along with their corresponding PWM inputs in order to generate a linear mapping between desired control surface angle and PWM value. To generate the desired deflections, we integrate angular velocity commands from the controller.

The experiments were performed in an 8m x 8m Vicon capture area, with 10 Vantage V5 cameras distributed along the periphery of the space. The Vicon system was configured to report position and quaternion data at 200 Hz to the computer running the planning and control algorithms, which was a Dell Precision 5530 laptop with an Intel i7-8850H CPU. The planning computer used the Robot Operating System (ROS) package "vicon_bridge" to generate a ROS topic with timestamped pose data, which was then differentiated over a single timestep to obtain linear and angular velocities. In addition, the quaternion data were transformed into yaw-pitch-roll Euler angles.

In order to deliver desired control inputs to the plane, a serial interface was developed to communicate with an 8 MHz Arduino Pro Mini microcontroller. The planning computer sends the desired PPM signal to the microcontroller, which relays this signal to a Spectrum DX6i transmitter module through the trainer port. The transmitter then broadcasts the signal to the RC receiver on the plane.

5.2. System Identification

To account for discrepancies between the modeled dynamics and the actual plane dynamics, we compared linear and angular accelerations predicted by the dynamics model with actual measurements from flight experiments. Modification of a few parameters was found to significantly improve the accuracy of the model. Specifically, we manually scaled the wing and horizontal fuselage area by a factor of 1.7, scaled the rudder area by a factor of 0.75, and set the backwash velocity coefficients to $\gamma_{ar} = \gamma_{al} = \gamma_r = 0.1$ and $\gamma_e = 0.3$. Here $C_{b_d} = 0.0$. In addition, we estimated the linear thrust model coefficients as $a_t = -4.9167$ and $b_t = 9.6466$. The results of these model adjustments can be found in Figures 7 and 8.

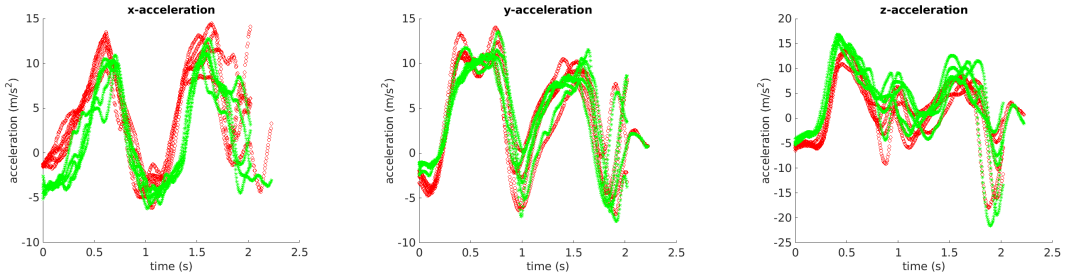


Figure 7. Plots show the linear accelerations predicted by the model and those recorded during the control trials. Red circles are the accelerations predicted by our model. Green asterisks are the recorded data.

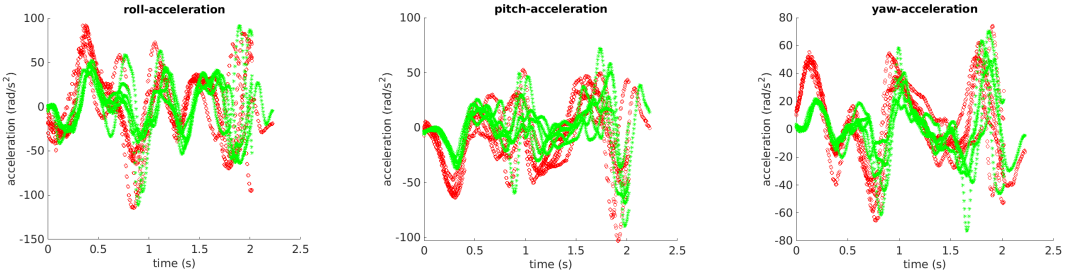


Figure 8. Plots show the angular accelerations predicted by the model and those recorded during the control trials. Red circles are the accelerations predicted by our model. Green asterisks are the recorded data.

5.3. Control Experiments

The experiments were conducted with a virtual map consisting of a system of hallways approximately 1.75 m wide. A goal point was specified in the hallway directly adjacent to the initial conditions of the plane, and several variations of the experiments were conducted. The trajectory optimizer was run at 5 Hz and the feedback controller was run at 50 Hz. For one set of trials, the launcher was used to provide a consistent set of initial conditions, and the controller was set to execute the full receding-horizon control stack. In another variant, the stack was executed through the receding-horizon trajectory optimization step, but the input trajectories were commanded open loop, without the feedback term generated from TVLQR. Additionally, several handthrown trials were performed using the full control stack.

Setting the final state constraints on the trajectory optimization problem to $\boldsymbol{\delta_f} = [0.1, 0.1, 0.2, 0.5, 1, 0.2, 100, 100, 100, 100, 100, 3, 3, 0.5, 2, 2, 2]$ was found to produce a reasonable balance between feasibility and desired behavior. The diagonals ($\mathbf{q}$, $\mathbf{r}$, $\mathbf{q_f}$) of the cost matrices $\mathbf{Q}$, $\mathbf{R}$, and $\mathbf{Q_f}$ for the local feedback controller were set to

$$\mathbf{q} = [25, 25, 25, 50, 50, 50, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2],$$
$$\mathbf{r} = [0.1, 0.1, 0.1, 25],$$
$$\mathbf{q_f} = [100, 100, 100, 100, 100, 100, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1].$$

A time horizon of $T_H = 1s$ was chosen as the minimum value that allows planning with sufficient forethought to execute complex maneuvers. For the initial trajectory in these experiments, this horizon corresponded to a a path length of $\approx$4–5 meters. The replanning frequency was chosen as 5 Hz based on the data from Figure 5.

5.4. Results

The testing of our direct NMPC algorithm in hardware demonstrated very successful results. For the task where we navigated in a virtual corridor, 10 out of 10 trials navigated without colliding with our virtual walls (see green lines in Figure 9). At the first corner, the highest angle-of-attack was 38° and 66° at the second corner. We also tested our algorithm without local feedback between plans. These trials almost immediately failed (see red lines in Figure 9). While we did occasionally violate our constraint towards the end of the trials (0.55 m radius), the actual plane never collided (Figure 10). We believe this is acceptable since SNOPT is able to relax these constraints during

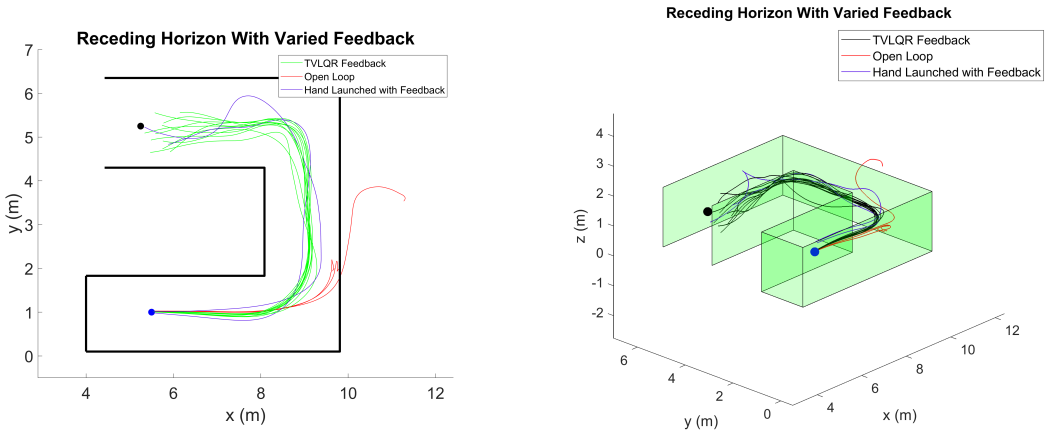


Figure 9. Left figure shows a top-down plot of the receding horizon trials with (green) and without local feedback (red) and handlaunched trials (blue). The blue and black points show the nominal start and goal conditions, respectively. All receding horizon trials with local feedback are successful while the trials without feedback fail. Right figure shows the trials in three-dimensions.

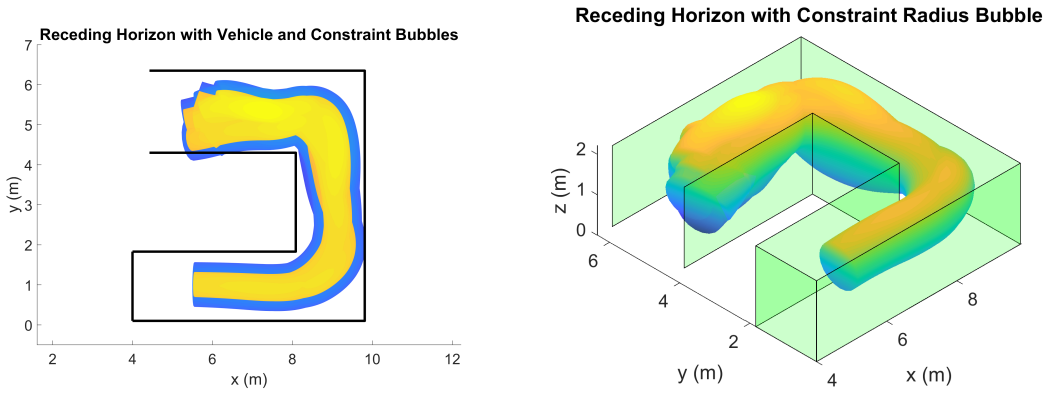


Figure 10. Left figure shows a top-down plot of the vehicle and constraint radii from the receding horizon trials. The blue region shows the inflated constraint radius around the vehicle, while the yellow region shows the effective vehicle radius. Minor constraint violations are observed at the end of the maneuver, but the vehicle still remains collision free. Right figure shows the constraint radii in three-dimensions.

elastic mode operation to minimize infeasibilities. To test robustness to initial conditions without a launcher, we also conducted some handthrown trials and had good success (Figure 9).

6. Outdoor Experiments with Onboard Processing

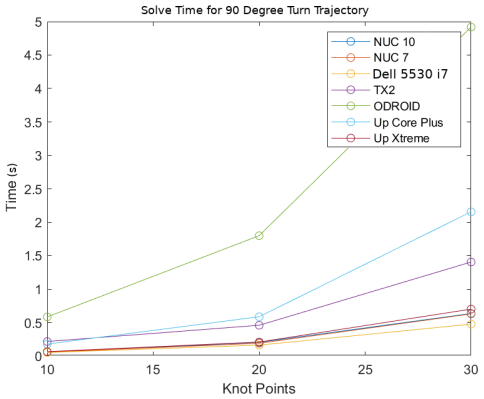
While the results of our indoor experiments were compelling, these experiments still did not address several important challenges. First, they did not address the issue of onboard computation. To be useful in more operationally-relevant environments, our NMPC algorithm must be able to run in real-time using an onboard processor. In our indoor experiments, we used off-board computation. While we did not use a particularly powerful processor, the processors associated with laptops are, in general, still significantly more capable than existing onboard processors. Second, our indoor experiments did not consider the impacts of external disturbances (e.g., wind) or state-estimation error, both of which can adversely impact controller performance. To explore these additional challenges we developed a more generalizable flight controller framework using the Robot Operating Sysem (ROS) and well as an associated onboard hardware stack, that together comprise the ACCIPITER autonomy stack.

6.1. Onboard Processor Selection

To select an onboard processor, we surveyed several available onboard processors commonly leveraged by small Group 1 UAS. We repeated the 90° turn warm-start trajectory generation analysis described in Section 5, but tested this on six candidate processors (see Figure 11). We also considered the mass of these processors in a COTS "usable" configuration, which in some instances (e.g., for the Nvidia TX2) required accounting for the weight of the required carrier board. We found that the Intel NUC preformed on par with the laptop computer we used for the indoor experiments, was comparable in mass to the Nvidia TX2, and was lighter than the UpCore Extreme. For this reason, and because we do not need parallel computing to run our algorithm, we chose the Intel NUC for our processing payload.

6.2. The ACCIPITER Software Stack

To generalize the approach developed in Section 3, facilitate interfacing with existing sensors, and support eventual multi-vehicle operations, we developed a modular UAV flight autonomy stack called ACCIPITER, or Aerobatic Control and Collaboration for Improved Performance in Tactical Evasion



Computer	Mass (kg)
Intel NUC 10	0.210
Intel NUC 7	0.200
Nvidia TX2	0.188
Odroid XU4	0.060
Up Core Plus	0.140
Up Core extreme	0.250

Figure 11. Plot on the left shows the required computation time to generate a post-stall 90° turn trajectory from a warm-start configuration. Table on the right shows how tested small form-factor processors compare with respect to mass in a "usable" configuration.

and Reconnaissance (see Figure 28). This autonomy stack leverages the Robot Operating System (ROS) and provides a ROS node for each major component required for flight control, including the controller, estimator, simulator, and hardware interface. ACCIPITER is designed to support the modeling, simulation, and control of electric-powered propeller-driven UAVs. To do this, we use the existing Universal Robot Description Format (URDF) provided by ROS to specify robot kinematics and dynamics and extend it to create the Universal Aerial Robot Description Format (UARDF). The UARDF includes specifications for aerodynamic surfaces and thrust sources to model the dynamics of aerial robots. Our UARDF also facilitates interfacing with the ACCIPITER hardware stack (see Section 6.3) by properly initializing the hardware interface node with the correct control surface and throttle configuration.

Because the autonomy stack is built using ROS, our framework naturally supports inter-process communication and allows sharing of information between ACCIPITER vehicles, such as state information or planned trajectories. The ACCIPITER autonomy stack also supports combining control modes with different cost-functions into a simple state machine. Transitioning to a different control mode, and thus a different cost function, is only permitted at the beginning of a planning interval so as to ensure the existence of a prior nonlinear programming (NLP) solution for warm starting.

6.3. The ACCIPITER Hardware Stack

The airframe selected for our outdoor hardware experiments was a 42-inch wingspan Edge 540XL airframe from Twisted Hobbys. Based on simulation results, we determined that this airframe would have sufficient payload capacity to support the Intel NUC processor. This airframe (see Figure 12) is also classified as an aerobatic aircraft and uses the same profile plane style design as our previous experiments (see Section 5). However, the wings are symmetrical airfoils instead of flat plates. For the powertrain, we used a Crack Series Pro 100 g 1050 Kv Motor, a Crack Series 45A ESC, and a 12x6 Wood Electric propeller. Four TWS1514D Servos were used to drive the aircraft's control surfaces. A 3 cell 2200 mAh lithium polymer battery provided power to the aircraft.

We chose to run our NMPC algorithm on an Intel NUC 10 high-level processor (HLP). Our HLP communicates over serial to a mRobotics Control Zero F7 autopilot running the Ardupilot stack (Team et al., 2022a). The mRobotics autopilot interfaces with the aircraft's powertrain and servos and also provides full-state estimates to the HLP, which it generates by using an Extended Kalman Filter (EKF) to fuse measurements from a 3-axis accelerometer, gyroscope, magnetometer, barometer, and airspeed sensor. The mRobotics autopilot also serves as a fallback autopilot in case of HLP failure and facilitates communication with the plane using the existing Ardupilot/Mission Planner framework. For position measurements, we used a uBlox ZED-F9P carrier Real-time

Figure 12. Photo on the left shows a view of the Edge540XL vehicle with onboard processing and a D455 depth camera. Photo on the right shows a close-up view of the Edge540XL vehicle and the key components for onboard sensing and control.

Kinematic (RTK) GPS unit and associated Helical Multiband Antenna. A RFD900+ telemetry radio was used to communicate with the base-station and provide RTK GPS corrections. For object detection and onboard mapping, an Intel Realsense D455 camera was rigidly attached to the fuselage behind the propeller.

Each control surface was connected to its associated servo via a four-bar linkage, which results in a nonlinear relationship between servo angle and control surface angle. To account for this nonlinearity, we generated a fourth-order polynomial mapping between control surface angle and servo PWM. To generate a PWM command, we integrate the desired control surface velocity to generate the desired control surface deflection and apply the polynomial mapping.

6.4. System Identification and Controller Configuration

As in the indoor experiments, we made some minor model adjustments to achieve better agreement with data. We scaled the wing area by a factor of 1.5 and set the backwash velocity coefficients to $\gamma_{ar} = \gamma_{al} = \gamma_r = 0.1$ and $\gamma_e = 0.3$. In addition, we estimated the linear thrust model coefficients as $a_t = -4.905$ and $b_t = 107.9$ using thrust stand data. We empirically found $C_{n_d} = 0.0705$. We also note that the Edge540XL has an airfoil, unlike the Edge540 which is a flat plate. Our model approximates the wing over the whole angle-of-attack as a flat plate. Therefore, we do not expect our aerodynamic model to be accurate in low angle-of-attack regimes, but rather underestimate lift. However, in practice, we found that the underestimation of lift did not seem to have an adverse effect on controller performance. We employed an collision radius for the optimizer of 0.95 m and a collision radius for the RRT of 1.0 m.

6.5. At-Altitude Tests in Simulated Urban Environments

We conducted a number of outdoor experiments with our 42-inch wingspan UAV. Initially, to test the control algorithm safely, we first conducted experiments at-altitude and used a simulated urban environment, where geometric obstacles were superimposed virtually in open space. For this, we leveraged a voxelized map of the Selby Combined Arms Collective Training Facility (CACTF) at Fort Benning. Instead of flying through a corridor as explored in the prior indoor motion-capture experiments, we explored a "building scanning" operation. In this mission, the fixed-wing UAV is tasked with flying around a specified building at high-speed with minimal stand-off distance in order to conduct visual inspection and reconnaissance. We prespecify the desired flight path around the building and plan to that flight path through the urban environment via a rapidly-exploring

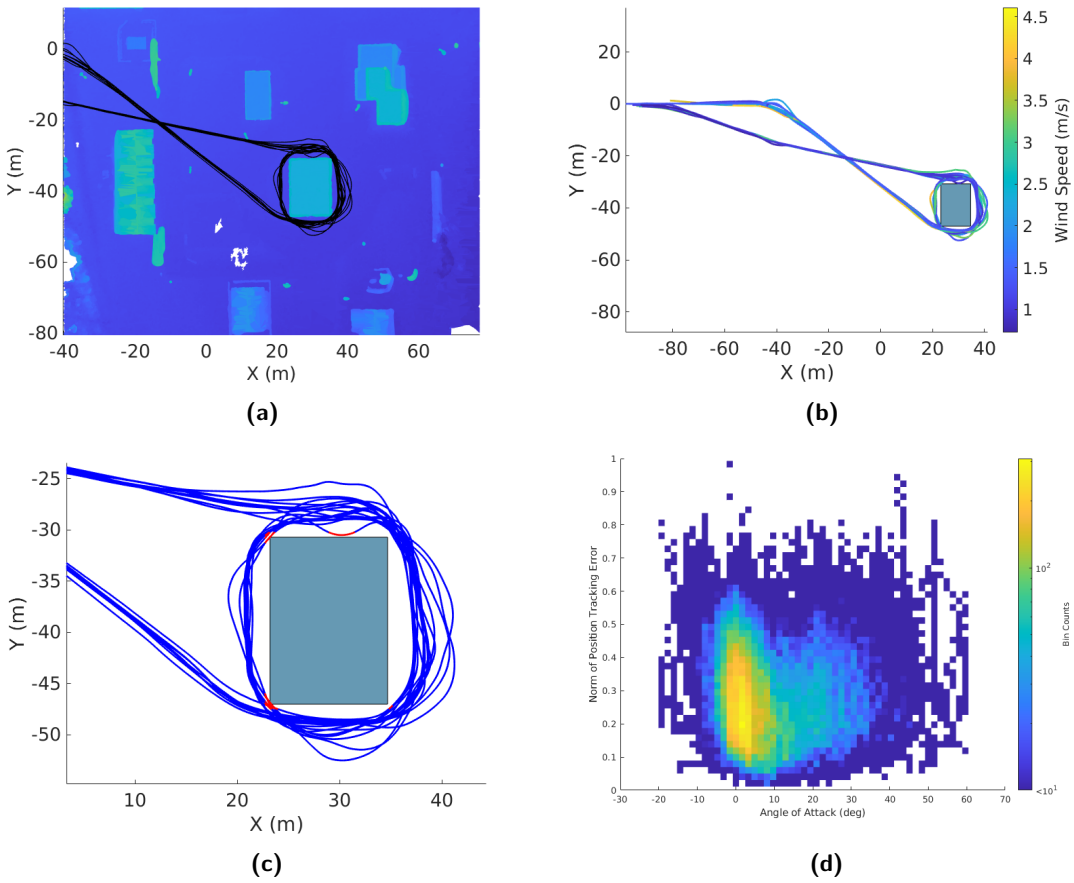


Figure 13. (a) Figure shows the flight position data (black) from nine runs of a building scanning operation in the Selby CACTF. (b) Figure shows the flight position data in the Seby CACTF where flight data are colored based on measured wind speed. (c) Figure shows the flight position data from ten runs of a building scanning operation in the Selby CACTF. Red-colored lines indicate collision with the building. 1/9 runs exhibited no collisions. (d) Figure shows total tracking error across vehicle angle-of-attack. The position tracking error is within 1m and is approximately the same magnitude across all angles of attack.

random tree (RRT). As part of the mission generation process, the user also has the ability to insert intermediate waypoints to achieve greater control over the RRT-generated flight path. As before, these flight paths provide a receding horizon point for the NMPC algorithm and the objective of the aircraft controller is to plan to this receding horizon point while not violating any collision constraints.

Our results demonstrated that our algorithm did scale to larger aircraft and demonstrated a tracking error of less than 1m across 10 trials (see Figure 13). In addition, our tracking error did not increase with angle-of-attack, indicating that a uniform tracking error exists across the nonlinear flight envelope. This performance was achieved by applying the same controller costs used in the indoor experiments, however, instead of using a terminal constraint on the receding-horizon state, we applied a terminal cost. We observed that a terminal cost afforded greater flexibility and provided improved performance when transitioning between controller types with different objective functions. Out of the 10 trials, 1 trial successfully made it around the building without collision. We initially designed the nominal building scan flight path with a two meter standoff distance. However, this proved to be inadequate, as it was not a significant enough standoff to prevent collisions at the building corners (see Figure 13), especially in the presence of wind. Since our controller only attempts to regulate the terminal state of the planned trajectory to the nominal flight path, the

earlier portion of the trajectory can be planned as close to obstacles as the optimization collision radius allows (0.95 m). With a tracking error of ≈1 m, this could easily lead to a vehicle collision, as it did in the at-altitude experiments. To overcome this issue, we explored increasing the standoff distance of the nominal RRT flight path, and in later experiments, also increased the collision radius used by the optimizer.

6.6. Flight Tests in Physical Urban Environment

Following our at-altitude tests, we conducted a number of flight tests in close proximity to physical buildings, as part of the Defense Advanced Research Projects Agency (DARPA) OFFensive Swarm-Enabled Tactics (OFFSET) field exercises. The first building we flew around was at the Joint Base Lewis-McChord CACTF. Based on our prior experiments, we increased our nominal building stand-off distance from 2 to 4 meters to prevent our vehicle from flying too close to the corners. Two of the three trials were successful (see Figure 14), with one of the trials clipping the corner of the building due to wind.

6.7. Wind Compensation

To address some of the poor performance observed during the DARPA OFFSET field exercise due to wind disturbances, we incorporated wind compensation into our controller. To do this, we leveraged

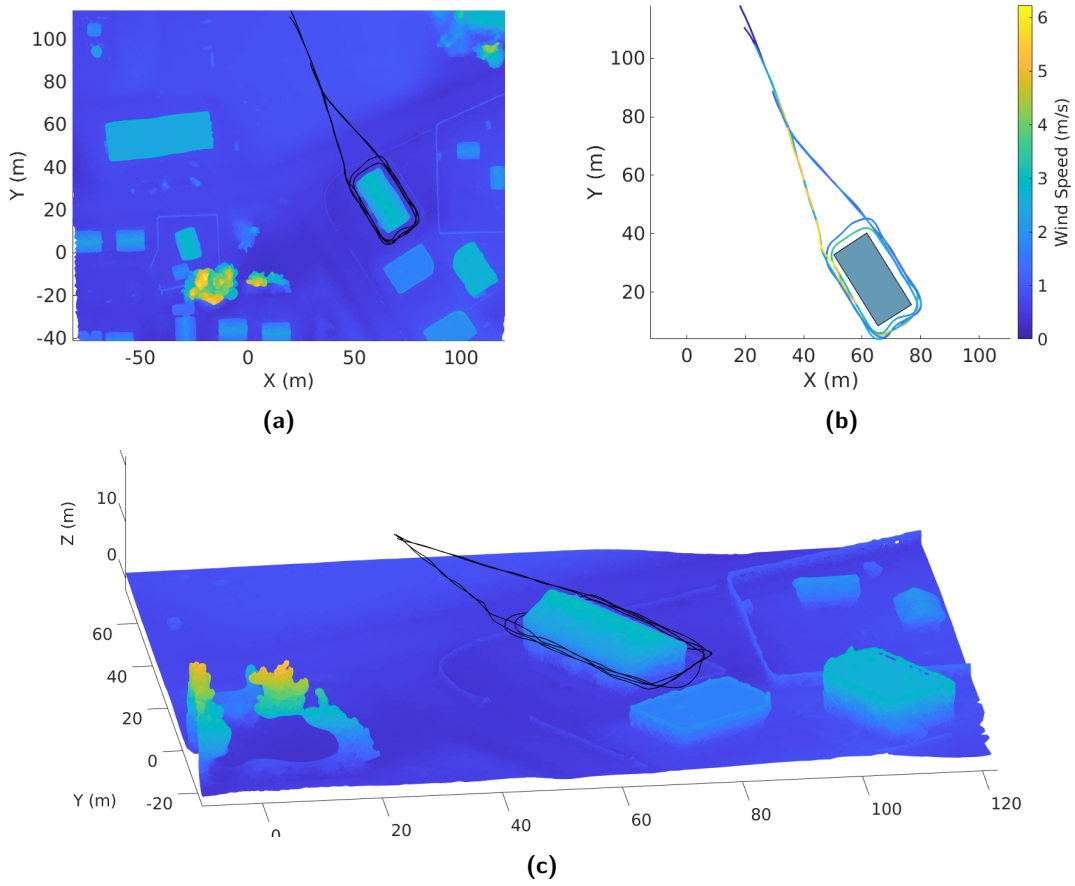


Figure 14. (a) Figure shows the flight position data (black) from two building scan tests in the JBLM CACTF. (b) Figure shows the how the wind varied during the experiments. (c) A 3D view of the flight trajectory followed by the vehicle.

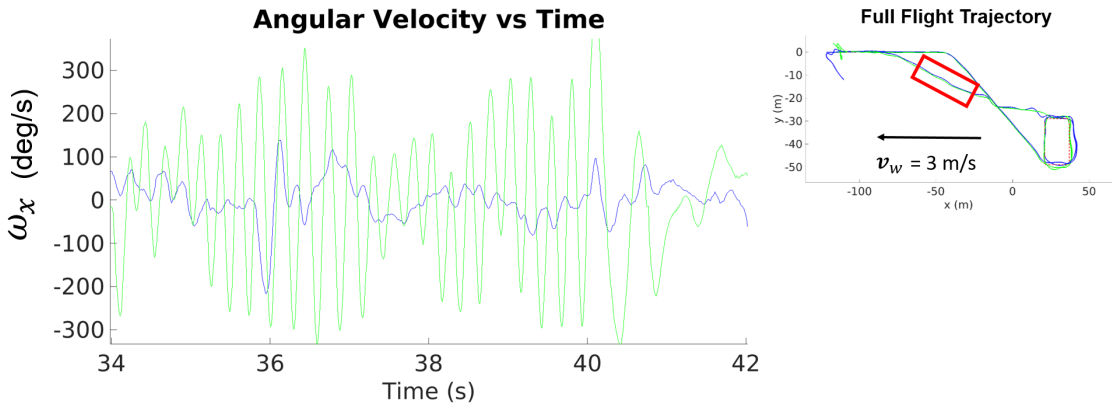


Figure 15. Figure shows two back-to-back flight tests with (blue) and without (green) controller wind compensation. In the presence of a $\approx 3\,\mathrm{m/s}$ headwind (red box), the wind compensation algorithm dramatically reduces controller oscillations.

the existing wind estimates provided by the Ardupilot state estimator. The Ardupilot EKF (Team et al., 2022b) fuses airspeed sensing and ground speed measurements to estimate the wind state. The covariance parameters of the EKF are tuned so that the wind estimates vary on the order of $\approx 0.1\,\mathrm{Hz}$. We incorporated these wind estimates into the vehicle dynamics model used for planning and control. Given how slowly the wind estimates change, we assumed that the wind is constant over the planning horizon ($\approx 1\,\mathrm{s}$). With wind compensation, we observed a significant reduction in the aircraft oscillations, especially in roll, that were previously noticeable in the presence of a headwind (see Figure 15).

7. Vision-Based Navigation and Feature Detection

After having demonstrated the ability to achieve agile fixed-wing flight with NMPC using an *a priori* map and RTK-GPS, we turned our attention to navigation using onboard sensing. To this end, we replaced our *a priori* occupancy grid map with NanoMap (Florence et al., 2018) for collision avoidance, while continuing to use RTK-GPS for position information. We also tested the detection of fiducial markers on the face of buildings while flying at high speeds and aggressive attitudes.

7.1. Mapping

For real-time map generation, NanoMap was chosen to replace OctoMap because it is more computationally tractable to query and to generate. Rather than creating an occupancy map, NanoMap maintains an asynchronous history of depth images and poses. The algorithm can then reverse search through this history to find a satisfactory depth image and return the k-nearest depth points to a query point. It is also able to propagate state uncertainty through a local history of depth measurements using the transformations between pose in its history.

A few adjustments were made to NanoMap to better suit our control strategy. Query replies were modified to include the transformation from the query body frame to the current body frame. This transformation is necessary for calculating the obstacle constraint gradients.

NanoMap query replies include information about the field of view status of the query point. This information specifies if the query point is in free space, occluded space, or outside of the field of view of depth measurements. If the query point was not within the field of view free space of any prior depth measurements, it had no valid corresponding depth measurement. In this case, the obstacle points are pulled from the most recent depth measurement. We modified NanoMap to return the K-nearest obstacle points from the most recent occluded field of view instead. This yielded more accurate nearest obstacle points, especially while planning through occluded space.

Algorithm 1. RRT(x_{init}, x_{goal}, Δx, map, K, Δt).

τ.init(x_{init}), $f \leftarrow [\,]$
for $k = 1$ *to* K **do**
  $x_{rand} \leftarrow$ RANDOM_STATE()
  $x_{near} \leftarrow$ NEAREST_NEIGHBOR(x_{rand}, τ)
  $u \leftarrow$ SELECT_INPUT(x_{rand}, x_{near})
  $x_{new} \leftarrow$ NEW_STATE(x_{near},u,Δt)
  if *PATH_OBST_FREE*(x_{near},x_{new}, *map*) **then**
    τ.add_vertex(x_{new})
    τ.add_edge(x_{near},x_{new},u)
    if *DISTANCE*(x_{new}, x_{goal}) $\leq \Delta x$ **then**
      $x_{goal} \leftarrow x_{new}$
      break
    if *IS_FRONTIER_NODE*(x_{new}, *map*) **then**
      f.add(x_{new})
if x_{goal} *not in* τ *or IN_UNKNOWN*(x_{goal}, *map*) **then**
  $x_{goal} \leftarrow x_f \leftarrow x_{goal}$.FIND_CLOSEST($f$)
Return τ, x_{goal}

7.2. Dynamic RRT Generation

In Section 5, a rapidly-exploring random tree (RRT) was used to generate global paths to the goal for horizon point selection. The RRT was expanded until the goal point was connected to the tree, and the path from the current position to the goal position was extracted. This path was iteratively pruned, smoothed with Bézier curves, and parameterized in terms of time. We refer to this final smoothed path as the smoothed RRT path, and it is used to select a receding horizon point for trajectory generation.

This strategy poses a problem for planning in unknown environments; the horizon point may be in or behind an unobserved obstacle. This is undesirable since the direct trajectory optimization problem is warm started with the previously generated trajectory. If a previously generated trajectory becomes intersected by newly obtained map data, the optimizer will often be unable to find a feasible solution.

To avoid this, the RRT path is constrained to known regions of the map using a method similar to that proposed in (Umari and Mukhopadhyay, 2017). Our method is outlined in Algorithm 1. As the tree expands, a list of frontier nodes (which exist in unknown space and connect to nodes in known space) is maintained. A node is in a known region of the map if the NanoMap query returns a field of view status indicating that the query point is in free space. If the goal point is not connected to the tree, or if the goal point is in an unknown region of the map, then the frontier node closest to the goal position is used as the goal instead.

At each control iteration, a truncated version of the prior RRT path is used to initialize the RRT. This truncation takes into consideration both the updated vehicle state and any newly detected obstacles. The beginning of the RRT path is truncated to minimize the distance between the start of the path and the vehicle state; the end is truncated to ensure that the path does not intersect with obstacles.

Reuse of the prior path reduces RRT computation time and also provides a more gradual evolution of the RRT path. This leads to a more incremental update of the receding horizon goal point and a reduced computational burden for the warm-started trajectory optimization routine.

7.3. Collision Constraints for Trajectory Optimization

In (Polevoy et al., 2022), we explored several Nanomap collision constraints in simulation. We found that a distance-to-obstacles collision constraint with noise compensation via covariance inflation

performed nearly as well as a probability of collision constraint with a lower computational cost. This constraint can be formulated as $d(\mathbf{x}) \geq r$, where r is the obstacle radius. Given the query, $\mathbf{x}$, and the $K = 10$ nearest neighbors $\mathbf{a}_i$, the distance to each nearest neighbor is calculated as $d(\mathbf{x})_i = \|\mathbf{x} - \mathbf{a}_i\|$. The distance to obstacle constraint is calculated as the average distance to the nearest neighbors $d(\mathbf{x}) = \frac{1}{K}\sum_{i=1}^{K}(d(\mathbf{x})_i)$. Similarly, the gradient for this constraint is calculated as $\nabla d(\mathbf{x}) = \frac{1}{K}\sum_{i=1}^{K}\left(\frac{\mathbf{x}-\mathbf{a}_i}{\|\mathbf{x}-\mathbf{a}_i\|}\right)$.

7.4. Experimental Setup

To test the use of onboard vision for collision-free navigation, we considered flight around a large building surrounded by trees. The test mission for the UAV was to fly towards two waypoints, both of which were obstructed by a large building. This environment showcases both straight, unobstructed zones as well as static obstacles. Navigation to the waypoints requires the system to make tight turns between the building and trees, which is similar to the hallway environment explored in Section 5.3. A mission was considered successful if the fixed wing was able to navigate around the building.

7.5. Simulated Perception Navigation Experiment

To evaluate our method using perfect perception data, we first simulated the depth camera data using Gazebo onboard the plane during a physical test flight. To do this, we provided the Gazebo simulation with an accurate mesh of the building and nearby light poles. As the fixed wing flew around the building, it sent odometry measurements to the simulator. In return, it received noiseless depth data given the camera field of view. The fixed wing used the distance to obstacles with standard deviation inflation with a 3 meter radius, a time horizon $T_H = 1s$, and a replanning frequency of 5 Hz.

The fixed wing was successfully able to navigate around the building to the waypoints using the simulated depth camera data. Figure 16 shows the fixed-wing's flight path overlayed with the obstacle distance constraint radius.

7.6. Vision-based Collision-Free Navigation Experiment

Given that our method worked on hardware with perfect perception data, we ran the same experiment with the same controller parameters using the depth data from the mounted D455 (see Figure 17). The depth images were down sampled by a factor of 10 and converted to point clouds for NanoMap.

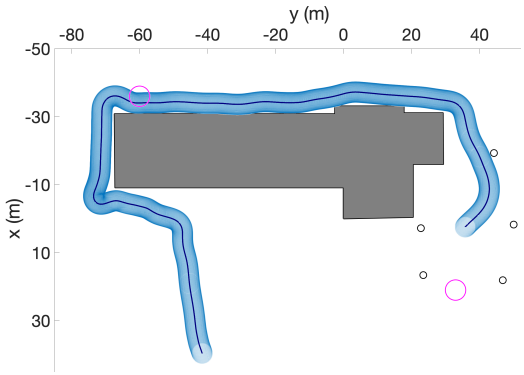


Figure 16. Plot shows the flight path taken by the Edge540XL vehicle around a physical building using a simulated depth camera for navigation. The gray shaded region represents the building and the black circles represent the light poles.

Figure 17. Screenshot of aerobatic fixed wing executing a post-stall maneuver (left) and corresponding depth measurement data (right). The global path plan is shown in red and the current trajectory is shown in orange.

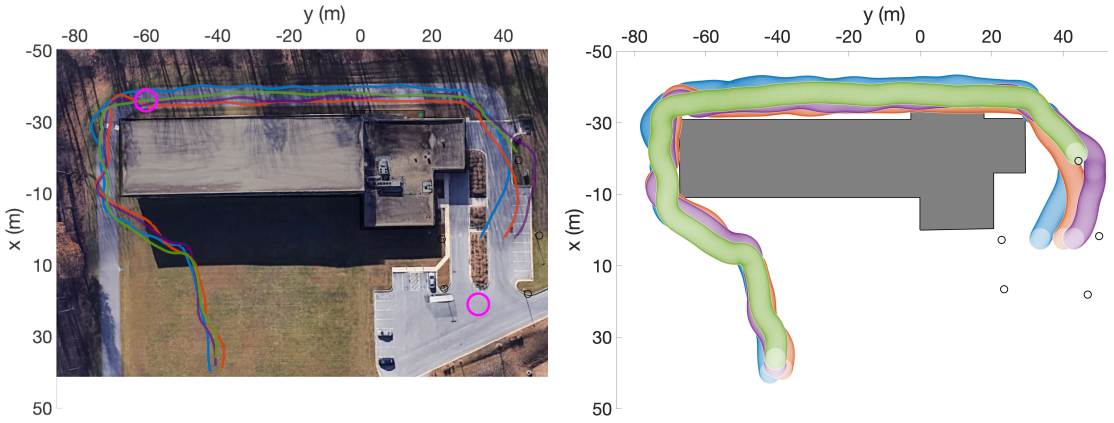


Figure 18. (Left) An overhead view of four flight paths taken by the Edge540XL vehicle around a physical building using the onboard RealSense D455 stereo depth camera for navigation. (Right) An overhead view of the four flight paths taken by the Edge540XL vehicle with the desired collision radius superimposed to highlight constraint satisfaction performance.

Out of six hardware trials, four successfully rounded the building. In one of these trials, the fixed wing crashed into a light pole after making it around the building. This crash was caused by the inability of the depth sensor to see the pole until the fixed wing was too close to react, possibly due to the relatively small size of the pole.

The paths of the successful trials are shown in Figure 18 along with an overlay of the obstacle distance constraint radii. The search depths of NanoMap queries during a turn are shown in Figure 19. While the majority of the depth queries were from the most recent pointcloud, the optimizer does query quite far back in the NanoMap history, motivating the need for a depth image history during aggressive maneuvers. The fixed wing achieved cruise speeds of 20 mph in straightaways and leveraged post-stall maneuvers during sharp turns (see Figure 20).

Two of the hardware trials failed, crashing into the building shortly after turning corners, as shown in Figure 21. In both of these failure cases, the RRT paths caused the fixed wing to turn towards the building very late while rounding the corners. The depth camera does not observe the part of the wall after the corner, causing new RRT paths to be generated through the apparent gap and resulting in crashes.

7.7. Feature Detection at High Speeds and Aggressive Attitudes

We also explored the possibility of achieving object detection at high speeds and unconventional attitudes with our onboard stereo camera at a 4 m standoff distance from a building. We placed both

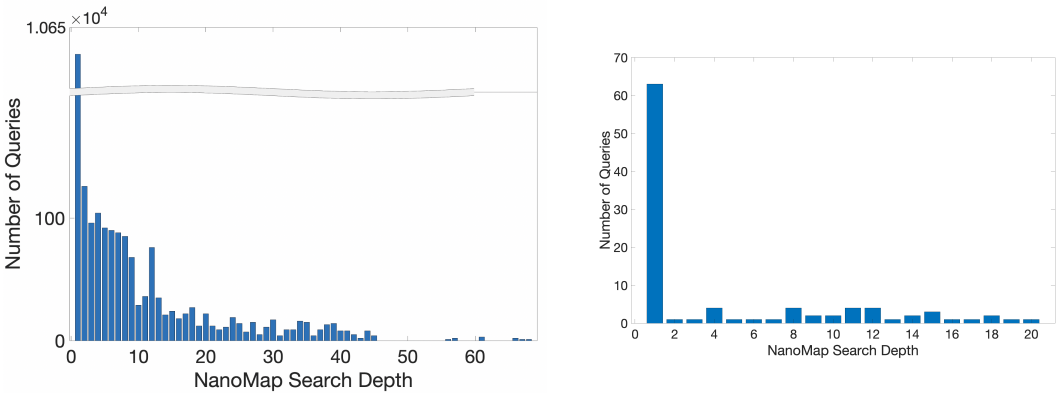


Figure 19. (Left) NanoMap search depth queries across all iterations for a hardware trial. Note that the number of queries for the current field of view is quite high, as demonstrated by the break in the vertical axis. This is likely due to the long straightaway distances. (Right) NanoMap search depth queries for a hardware post-stall turn.

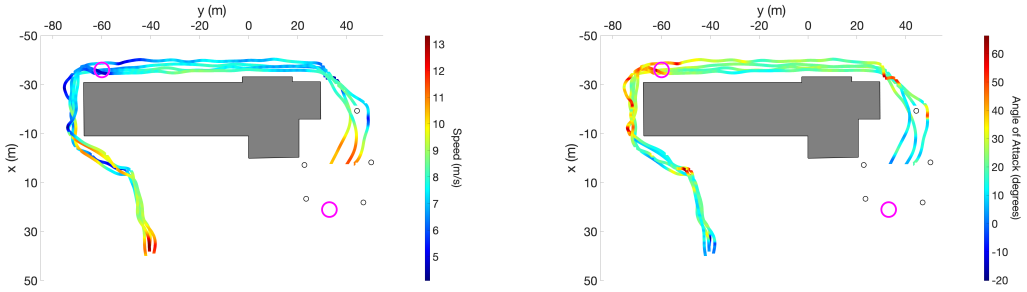


Figure 20. Speed and angle-of-attack achieved while executing vision-based collision-free navigation close to a building.

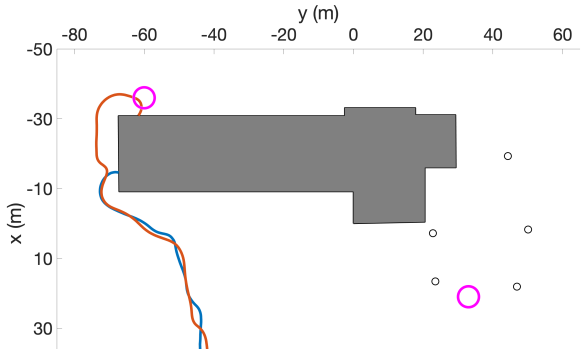


Figure 21. Plot shows the flight paths of the failed hardware trials. The pink circles denote the sequential goals. In each case, the vehicle executed an aggressive turn and did not receive sensor readings from the wall until it was too late to avoid a collision.

346x346mm markers and 125x125mm markers on the surface of a building, leveraged off-the-shelf AprilTag software (Wang and Olson, 2016) to process our camera images and detect the makers, and flew our aircraft around the building at typical flight speeds (12 m/s). Infrared and depth resolution was set to 1280x720 at a framerate of 30 FPS. We were able to detect all of the larger makers (4/4) and most of the small markers (3/4), even when at high speeds and aggressive pitch angles (see Figure 22). We also found good agreement between the estimated maker positions and

Figure 22. Stills of successful feature detection while flying close to a building at high speeds and aggressive attitudes.

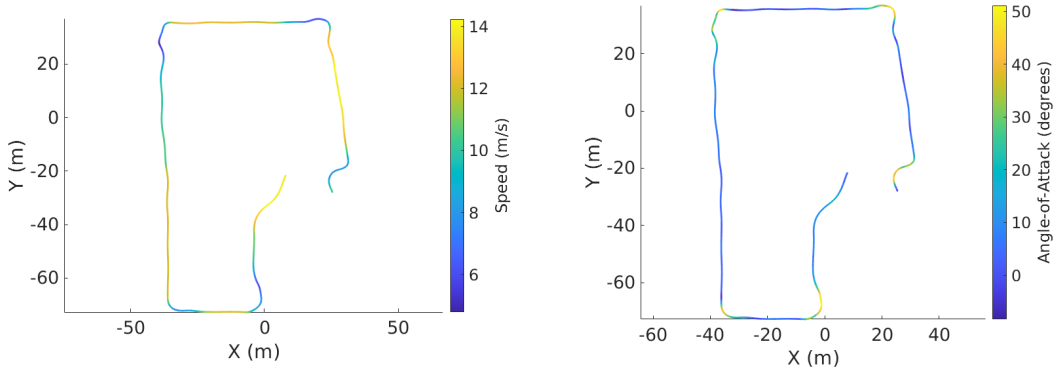


Figure 23. Figures show the speed of the aircraft (≤ 30 mph) and the angle-of-attack ($\leq 50^\circ$) during the feature detection experiments.

the approximate positions of the markers on the building. Figure 23 shows the range of speeds and angles of attack measured.

8. Swarm System Integration for Urban Operations

Our ultimate objective is to deploy our aerobatic fixed-wing UAVs as part of a heterogeneous swarm operating in an urban environment. While the planning and control approach outlined in Sections 3–7 is critical to navigating in urban environments, it does not address several important aspects of swarm system integration. First, we must address the challenges with the launch and recovery of fixed-wing UAVs. To do this, we leverage our NMPC algorithm, along with a simple hardware modification, for post-stall take-off and landing. Second, we must enable multi-UAV collision avoidance at high speeds; we employ our NMPC algorithm to achieve this as well. Finally, we must be able to plan and deploy our vehicles as part of a large heterogeneous swarm. To do so, we leverage Command and Control of Aggregate Swarm Tactics (CCAST) (Clark et al., 2021), an existing swarm framework designed for urban operations and used in the DARPA OFFSET (Chung, 2021) program.

Figure 24. The photo on the left shows the six aerobatic fixed-wing vehicles fabricated to support multi-vehicle operations. The still on the right shows an instance of three aerobatic fixed wings flying in close proximity during OFFSET Field Exercise 6.

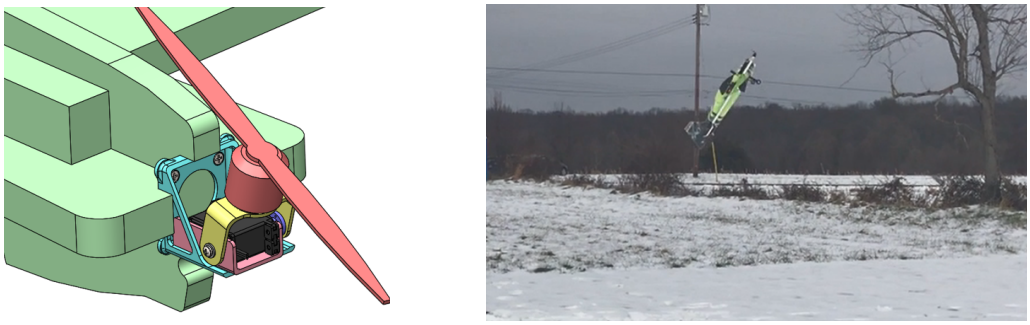


Figure 25. The rendering on the left shows a CAD model of the simple gimbal mechanism used for automatic take-off. The still on the right shows the aerobatic fixed-wing UAV taking off from rest.

8.1. Automatic Take-Off

Oftentimes, fixed-wing UAVs must be hand-launched or launched from a catapult. This lack of vertical take-off capability leads to a number of logistical challenges when conducting swarm operations. Not only would significant human intervention be needed to launch the vehicles, but often large deployment areas with specialized hardware are required. To mitigate these logistical challenges, we again decided to take advantage of post-stall flight, this time during take-off in a manner similar to (Peloquin et al., 2016; Moore et al., 2018). Do do this, we added a simple active gimbal mechanism to take advantage of our high thrust-to-weight ratio for automatic take-off (see Figure 25). By adding a single servo at the nose of the aircraft, we were able achieve effective thrust-vectoring with a single propeller for rapid takeoff with minimal added weight ($\approx 155\,\mathrm{g}$). A preliminary analysis of the vehicle's simplified dynamics showed that a minimal launch duration ($\approx 1\,\mathrm{s}$) was possible once the motor tilt angle reached 60°.

Upon activation, the take-off controller begins spinning the propeller at low speed and commands the gimbal to its initial desired angle of 85° above the horizon. After a short delay, the motor is commanded to use full power and, as the plane pitches up, the gimbal is controlled to maintain a constant 85 degree pointing angle relative to the horizon. Once the plane exceeds a specified pitch angle of 55° (at which the aircraft has a slightly positive vertical thrust-to-weight ratio), the desired gimbal angle is reset to 55° above the horizon. A Linear Quadratic Regulator (LQR) is used to generate control surface commands which stabilize the plane in a 55 degree climb. Once the aircraft reaches a particular altitude, it uses the NMPC algorithm and transitions into forward flight.

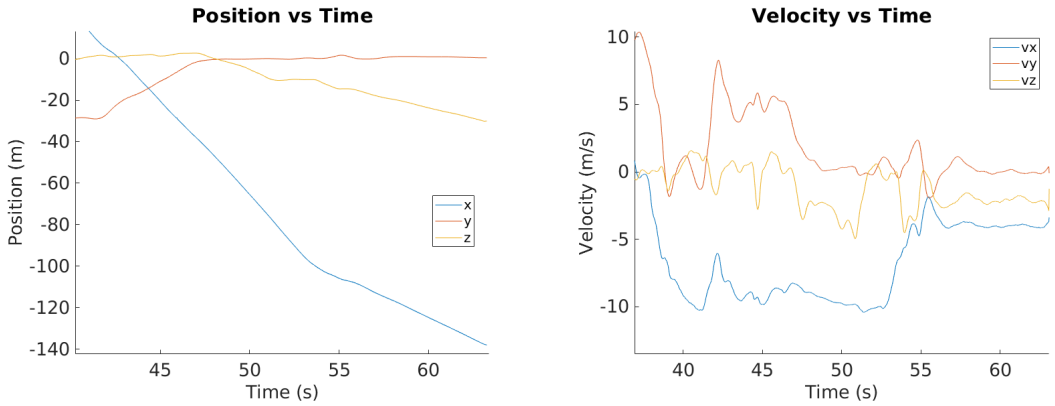


Figure 26. The plot on the left shows the position of the vehicle versus time during the landing maneuver. The objective is to land (−140, 0, −30). The plot on the right shows the velocity of the vehicle versus time during the landing maneuver. The vehicle is able to reduce forward speed to 4 m/s and downward speed of 2 m/s at touchdown.

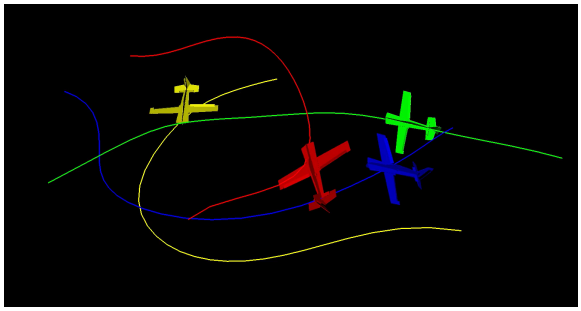


Figure 27. A still from a simulation of four collision-free trajectories optimized using the shared trajectory obstacle map constraints leveraged by the NMPC algorithm

8.2. Automatic Landing

Similar logistical considerations exist with regards landing. Manually landing fixed-wing aircraft, or executing landings that require long, flat runways will not scale to swarm scenarios. Therefore, we leveraged our same receding-horizon NMPC approach to execute precision deep-stall landings. Our approach is similar to (Mathisen et al., 2016; Mathisen et al., 2021), but is able to optimize full three-dimensional trajectories for a receding-horizon deep-stall maneuver in real-time. To achieve these landings, the aircraft plans a landing approach to eliminate the tail-wind component. The aircraft then executes a post-stall maneuver to transition into a high angle-of-attack regime to slowly descend along a nominal flight path to a target landing site. With this controller, we were able to achieve touch-down speeds of 4 m/s in the longitudinal direction and 2 m/s down in the lateral direction (see Figure 26).

8.3. Multi-UAV Collision Avoidance

To prevent collisions between vehicles in the multi-plane scenario, a trajectory obstacle constraint was added to the nonlinear optimization problem being solved online by each aircraft (see Figure 27). UAVs share trajectories with the team and each vehicle maintains a "trajectory obstacle map" that characterizes all the planned future positions of its team members. For every trajectory in the trajectory obstacle map, a spherical collision object was associated with each knot point in the trajectory. The trajectory optimizer queries the trajectory obstacle map during every planning interval to

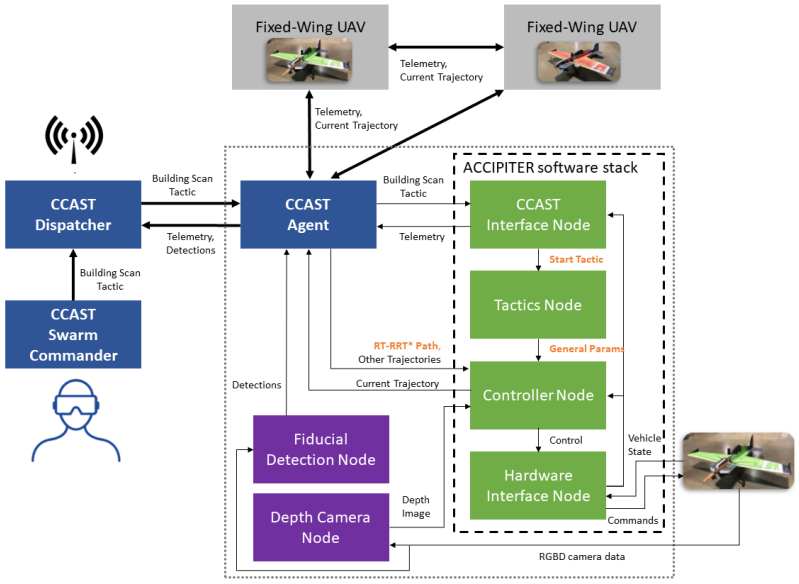


Figure 28. A diagram of the ACCIPITER technology integrated into the CCAST architecture. The dark dashed lines denote the core ACCIPITER capability. The green and purple blocks are run on the aerobatic aircraft. The orange text denotes features that have not yet been integrated, but could facilitate a more cohesive integration with the CCAST swarm.

evaluate a minimum distance constraint and associated gradient. For collision avoidance, we employ a decentralized approach, where each vehicle plans online asynchronously based on the current known set of trajectory obstacles. In practice, we observe that the vehicles can replan fast enough to ensure collision-free flight, even though they do not solve a joint trajectory optimization problem.

8.4. Swarm System Integration

To support large-scale swarm operations, we integrated our ACCIPITER autonomy stack with the CCAST framework (Clark et al., 2021). Our vehicles were able to receive mission commands from the CCAST Swarm Commander and share information on an LTE network via a ROS interface (see Figure 28). To connect to the LTE network, we integrated an LTE modem onto our aircraft. To facilitate task allocation, aerobatic tactics were prespecified based on an *a priori* map of the urban test site. Typically, these tactics included a precomputed flight path through the urban environment. This precomputed flight path was then used to set up the receding-horizon goal point for the NMPC algorithm. For example, for a building-scan tactic, the flight plan was comprised of a path that connected the launch site to the building of interest. Instead of specifying the path itself, the Swarm Commander could assign a particular building scan mission to a particular plane at run-time. When a fixed-wing UAV detected a fiducial marker, the detection and marker position was sent back across the LTE network to the base-station and displayed on a map of the urban environment. Planned trajectories for aerobatic fixed-wing team members were passed over the LTE network and stored in each aircraft's trajectory obstacle map. Only aerobatic vehicles were included in the trajectory obstacle map, as it was reasoned that the other systems would be moving slowly enough compared to the fixed-wing UAVs that using current telemetry information for collision avoidance would be sufficient.

8.5. Swarm System Integration Experiments

We conducted a number of hardware experiments to test integration with the CCAST swarm system. We first conducted an at-altitude building scanning operation with two aerobatic vehicles. We tested

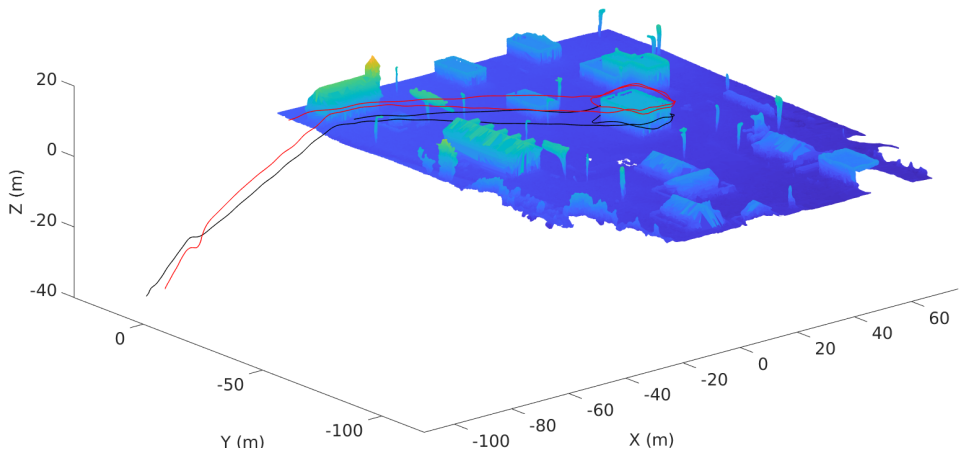


Figure 29. A position plot of two aerobatic fixed-wing UAVs (red and black) executing an automatic take-off and building-scan operation in the Selby CACTF during an at-altitude trial with virtual obstacles. The vehicles use the gimbal mechanism to take-off and then use the NMPC algorithm to transition into forward flight. The planes were also exchanging trajectory information via Raytheon BBN's CCAST software for local collision avoidance.

launching the vehicles from the CCAST dispatcher, sharing trajectories and state information over the CCAST network, and UAV collision avoidance. The vehicles successfully launched in close proximity and executed the two-plane building scan (see Figure 29). The nominal flight plans for each vehicle were designed to be spatially separated by at least 4 m to ensure that the nominal flight paths were collision free. We did not use wind compensation during this trial, and this led to some pitch oscillations.

We also tested our swarm system integration during DARPA OFFSET Field Exercise 6. In this scenario, the aerobatic fixed-wing UAVs were used as a fast, preliminary reconnaissance asset, and they were not used concurrently with other slower moving autonomous vehicle team members. The nominal flight paths were preplanned and designed to be spatially separated in altitude by 4 m. We demonstrated successful navigation around a building during two trials (see Figures 30 and 31). The tactics were assigned and the aircraft was launched by the Swarm Commander via the Swarm Tactic Operations Mission Planner Graphical User Interface (STOMP GUI). During the flight, fiducial markers were detected and registered on the map by the UAV and sent back to the swarm base station (see Figure 34). We were also able track our vehicle in the CCAST virtual reality interface (Immersive Interactive Interface, or I3) (see Figure 34).

We then demonstrated flight through a narrow urban corridor with a ≈ 5 m width (see Figures 32 and 33). Our vehicle navigated through this corridor four times during the field exercise. The level of wind in the corridor had a significant impact on how much the vehicle deviated from its nominal flight path. Finally, we demonstrated the autonomous launch and navigation of three aerobatic fixed-wing UAVs into the CACTF (see Figure 24).

8.6. Future Steps for Swarm System Integration

Further integration with the swarm system could be achieved by using CCAST's RT-RRT* (Naderi et al., 2015) global planner in place of the RRT used by the ACCIPITER stack. Not only does the tree rewiring in RT-RRT* lends itself to smoother flight paths, but it also affords tighter integration with the CCAST task allocation algorithms. Such an integration would greatly improve our ability to generate missions on the fly. Leveraging the RT-RRT* planner would also automatically ensure the high level flight path avoids other vehicles in the swarm; this would provide a layer of airspace deconfliction in addition to the more reactive avoidance provided by NMPC algorithm. We could

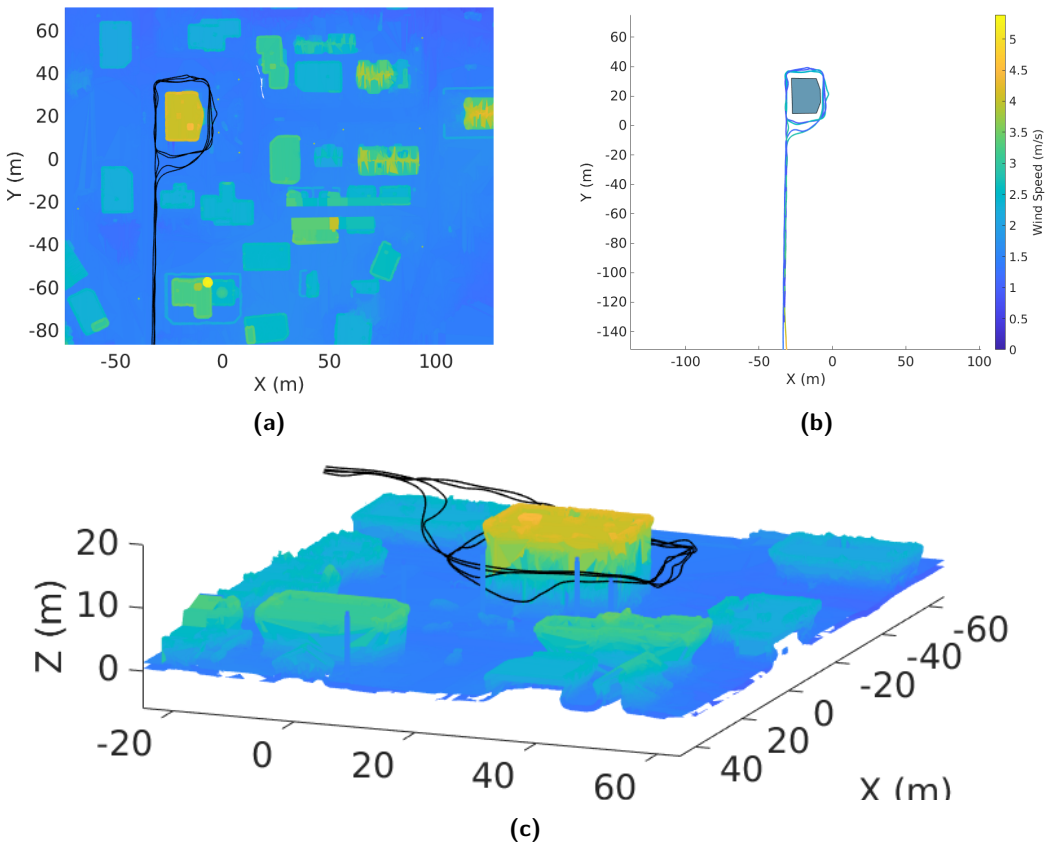


Figure 30. (a) Figure shows the flight position data (black) from two building scan tests in the Cassidy CACTF. (b) Figure shows the how the wind varied during the experiments. (c) A 3D view of the flight trajectory followed by the vehicle.

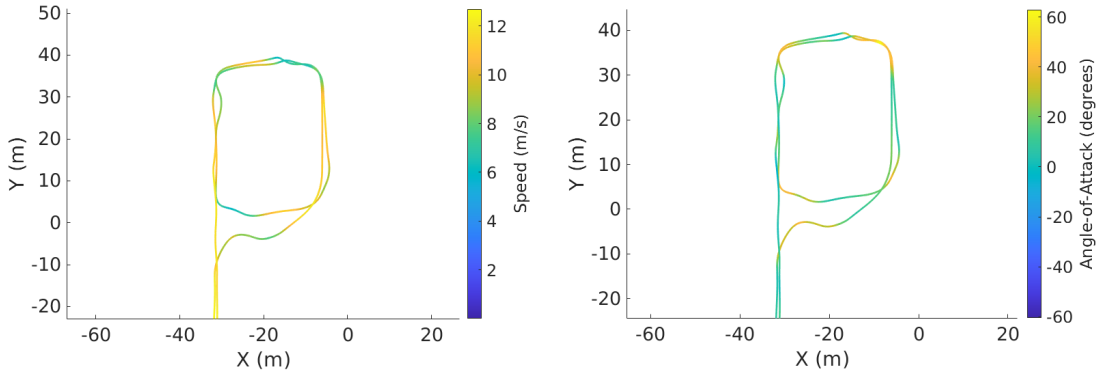


Figure 31. Figures show the speed of the aircraft (max 30 mph) and the angle-of-attack (max 60°) while navigating around Building 15 in the Cassidy CACTF.

also apply additional constraints to the online trajectory optimizer to restrict all trajectories to be within a prespecified radius of the flight path. Initially, we did not believe that the RT-RRT* approach, which updates at 1 Hz, would replan fast enough to support aerobatic fixed-wing vehicles. However, given the length scale of urban environments (> 100 m), in a known environment it may be possible to plan flight paths far enough into the future to enable agile fixed-wing flight. For handling

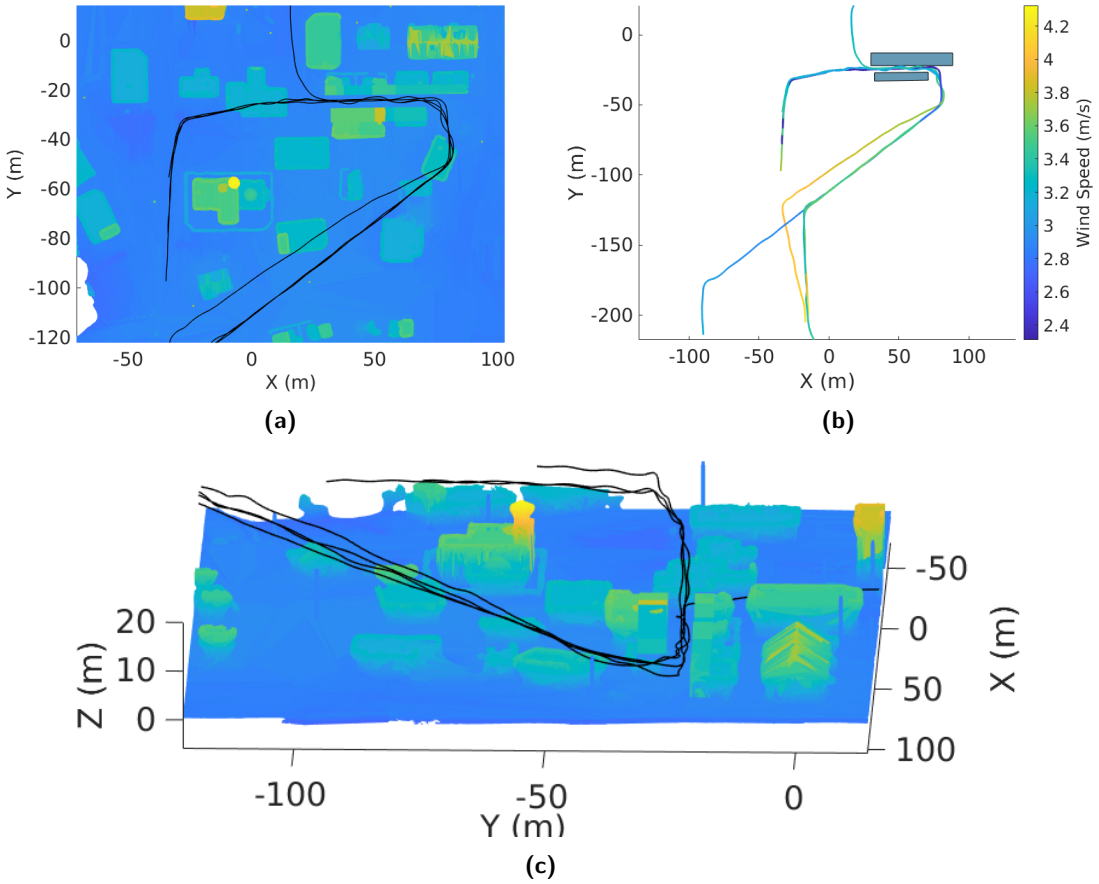


Figure 32. (a) Figure shows the flight position data (black) from two corridor navigation tests in the Cassidy CACTF. (b) Figure shows the how the wind varied during the experiments. (c) A 3D view of the flight trajectory followed by the vehicle.

Figure 33. Stills of the fixed-wing UAV navigating in the Cassidy CACTF.

unmapped obstacles or dynamic obstacles in the environment, we could leverage onboard sensing and our vision-based planning algorithm.

9. Discussion

In this paper, we described an approach for real-time receding horizon control with fixed-wing UAVs that can exploit post-stall aerodynamics to make tight turns in constrained environments. We demonstrated successful performance of this algorithm on hardware in indoor environments using

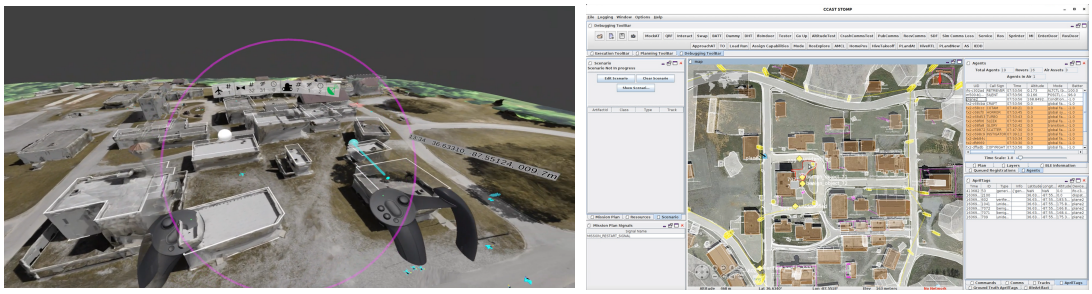


Figure 34. (Left) View of the I3 virtual reality interface (Clark et al., 2021) during the Cassidy CACTF corridor flight. (Right) View of the STOMP GUI during a building surveillance operation. The yellow flower-shaped icons denote a detected fiducial marker.

offboard sensing and processing, in outdoor urban environments using onboard processing, and in outdoor environments while using stereo vision for collision-free navigation. We also demonstrated that our algorithm can be formulated to support other important multi-vehicle operations, including automatic take-off, landing, and vehicle-vehicle collision avoidance.

During our many experiments and field tests, we made a number of observations. For the 42-inch wingspan aircraft, we observed that the worst-case tracking error seemed to be limited to about 1 m when using our NMPC algorithm and associated ancillary feedback controller. This was an important consideration when selecting parameters such as the stand-off distance for the nominal flight path and trajectory obstacle constraint. As mentioned above, in the outdoor experiments without vision-based navigation, we observed that a 4 m nominal stand-off of the nominal trajectory was sufficient to ensure good performance. In the case of the vision-based navigation experiment, we had the nominal flight path (RRT) collision distance set to 3 m and the trajectory obstacle constraint set to 2.95 m. This also proved to be a suitable parameterization to ensure collision-free flight.

An opportunity also exists to incorporate higher fidelity wind measurements into the controller. We observed that the wind estimates provided by the autopilot using the pitot tube and ground speed measurements were not sufficient to capture sudden gusts due to the large filter time constant applied to the wind estimates. In addition, the single-axis nature of the pitot tube meant that the wind measurements would be impoverished at high angles of attack. For this reason, we did not update the wind estimates in our controller when the vehicle was operating at high angles of attack. Future work will explore more sophisticated wind state estimators and novel sensors capable of measuring pressure or airspeed in three dimensions.

We also discovered that it was possible to achieve collision-free navigation using onboard sensing with a rigidly mounted stereo depth camera. By biasing the camera with a downward tilt, the vehicle was able to sufficiently detect obstacles at high angles of attack. One failure mode for this sensing paradigm is that, since environment perception is decoupled from control, it is possible to maneuver aggressively and not detect an obstacle until it is too late for the vehicle to correct. For instance, consider the case where the vehicle executes a turn around a corner of a building, and the building wall on the other side of the corner is outside the field of view. By the time the obstacle moves into the vehicle's field of view, the vehicle does not have enough time plan a new path before colliding. This failure mode could be remedied by an approach that more tightly couples trajectory optimization with perception or that leverages a neural network to predict the environment beyond the field-of-view as in (Katyal et al., 2021).

Another important observation is that the vehicle is capable of detecting fiducial markers while flying at very high speeds. This is a valuable result, because the fixed wing can be effectively used as a visual reconnaissance agent. Leveraging observability metrics (Grebe et al., 2021) to improve detection of or search for these fiducials is another promising area of future work.

One of the most important observations is the utility of implicit integration constraints as part of the trajectory optimization approach. Implicit integration constraints allow for both stable

numerical performance and improved trajectory tracking error for a fewer number of knot points. Furthermore, direct methods allow for a straightforward means of encoding collision avoidance via path constraints. The generalizability and flexibility of our NMPC approach is also quite promising. Not only does it work on a complex nonlinear system using an analytical dynamics model, but it can support a wide variety of different behaviors by shaping costs and constraints.

In the future, we plan to explore collaborative control of multiple aerobatic fixed wings in more depth. Not only is there an opportunity to explore dense swarming in hardware, but there is also an opportunity to explore applying machine learning to cost function generation to shape and create more advanced multi-agent behaviors. In addition, instead of just sharing telemetry information, there is an opportunity to use our onboard depth cameras to achieve vision-based vehicle-vehicle collision avoidance. Another future direction of this research will be working toward more rigorous guarantees of controller performance and safety, such as is being explored in (Manchester and Kuindersma, 2017; Bonzanini et al., 2019; Singh et al., 2019; Garimella et al., 2018). Reasoning about the statistics of controller performance during the trajectory optimization phase of our algorithm has the potential to considerably improve robustness in the presence of significant uncertainties.

Acknowledgments

We are grateful to the Defense Advanced Research Projects Agency (DARPA) OFFensive Swarm-Enabled Tactics (OFFSET) program and their support of the work presented in this paper.

Appendices

A. Analysis of Post-Stall Turns

To highlight the maneuverability (i.e., turning radius) advantages provided by post-stall flight, we explore the performance of aerial vehicles in a turn. To do this, we analyze a trim condition for a simplified dynamics model based on the dynamics in Section 3.1. By assuming direct control of the vehicle's Tait-Bryan angles, $\boldsymbol{\theta}$, zero vertical velocity, and restricting the motion of the vehicle to a circular arc, we can write the steady state trim condition as

$$c(\boldsymbol{\theta}, v, r, f_t) = \mathbf{R}_b^r \mathbf{f}_a^b + \mathbf{R}_b^r \mathbf{f}_t^b + \mathbf{g} - \mathbf{a} = 0. \tag{27}$$

Letting $\mathbf{e}_{x,y,z}$ be the unit vectors in the body fixed frame and $\mathbf{e}_{r,\theta,Z}$ be the unit vectors in polar coordinates along the turn, we write

$$\mathbf{f}_t^b = f_t \mathbf{e}_x \qquad \mathbf{g} = -mg\mathbf{e}_Z \qquad \mathbf{a} = -\frac{mv_\theta^2}{r}\mathbf{e}_r, \tag{28}$$

where $\mathbf{f}_t^b$ is the thrust force in the body fixed-frame, $\mathbf{g}$ is the gravity force, $\mathbf{a}$ is the centrifugal force, and $\mathbf{R}_b^r$ is the rotation matrix from the body frame to the world frame. f_t is the total thrust force, m is the vehicle mass, g is the gravitational constant, v_θ is the vehicle's transverse speed, and r is the radius of the turn. For a fixed wing, the body aerodynamic forces are derived according to flat plat theory as

$$\mathbf{f}_a^b = \rho S v_\theta^2 \sin(\alpha)\mathbf{e}_z, \tag{29}$$

where S is the wing area and ρ is the density of air. We approximate the quadrotor as a sphere with drag determined as

$$\mathbf{R}_b^r \mathbf{f}_a^b = \frac{-\rho S v_\theta^2}{4}\mathbf{e}_\theta. \tag{30}$$

Here S is the projected area of the sphere. For the low angle-of-attack fixed-wing flight regime, we write

$$\mathbf{R}_b^r \mathbf{f}_a^b = \rho S v_\theta^2 (C_L \mathbf{n}_L + C_D \mathbf{n}_D), \tag{31}$$

where $\mathbf{n}_L$ and $\mathbf{n}_D$ are the unit vectors in the lift and drag directions respectively. The lift and drag coefficients C_L and C_D come from a linear and quadratic approximation of the lift and drag coefficients for a NACA 2412 airfoil. We also add a constraint on the angle-of-attack, $\alpha \leq \alpha_{crit}$, where α_{crit} is the critical angle-of-attack.

For this simplified dynamics model, feasible trim conditions exist for a wide range of speed and turning radius values. To reduce the set of possible solutions, we restrict ourselves to the set of speeds and turning radii that minimize propulsive power.

This leads to solving the following nonlinear program that solves for v and r while minimizing propulsive power:

$$\min_{\boldsymbol{\Theta},v,r,f_t} \sum_k P_k$$
$$s.t. \quad c(\boldsymbol{\theta}, v_\theta, r, f_t) = 0, \tag{32}$$

where P is the power for the k^{th} rotor.

To evaluate the power consumed by a vehicle during a turn maneuver, we use the formulation for the aerodynamic power as derived in (Chauhan and Martins, 2020; Chauhan, 2020), given as

$$P = f_t V_\perp^\infty + \kappa f_t \left(-\frac{V_\perp^\infty}{2} + \sqrt{\frac{V_\perp^\infty}{4} + \frac{f_t}{2\rho A_{\text{disk}}}} \right), \tag{33}$$

where $V_\perp^\infty$ is the component of the freestream velocity perpendicular to the actuator disk, A_{disk} is the area of the actuator disk, and κ is a correction factor to deal with unmodeled losses. For our analysis $\kappa = 1.2$. We note that this formulation does not follow Glauert's modified momentum theory for forward flight (Glauert, 1927) as is done in other quadcopter modeling approaches (e.g., (Huang et al., 2009; Bangura and Mahony, 2017)), but rather assumes that only the free-stream velocity perpendicular to the actuator disks contributes to the generated thrust.

A plot of the velocity-turning radius relationship across a set of fixed-wing and quadcopter UAVs of different scales can be seen in Figure 1. The lines with a color gradient represent post-stall minimum power turns ($\alpha > 15°$). The dashed black lines represent fixed-wing UAVs restricted to the low angle-of-attack domain ($\alpha < 10°$). The solid red region are quadcopter minimum power turns. At high angles of attack, the fixed wing is capable of low turn radii maneuvers on par with the quadcopter UAV. A post-stall fixed-wing UAV is also able to achieve smaller turn radii than a fixed-wing UAV restricted to the low angle-of-attack regime, which must dramatically increase speed and bank angle to minimize a turn.

B. Energy Analysis

We would like determine when it is more advantageous to deploy an aerobatic fixed-wing UAV instead of other aerial systems. Even though fixed wings are significantly more energy efficient in steady-level flight than a quadcopter, they are in general, less efficient in hover. Understanding this trade-off as a function of environment is particularly important in the context of heterogeneous swarms since operators often desire to optimally allocate navigation tasks.

To characterize for what missions an aerobatic fixed-wing UAV operating at high angles of attack has an energetic advantage over a quadrotor, we apply an approximate first principles analysis of the energy consumed by a UAV navigating along a flight path decomposed into straight lines and turns.

We approximate the energy expended along a portion of the flight path as

$$E_{\text{total}} = E_{\text{straight}} + E_{\text{turn}}, \tag{34}$$

where $E_{\text{turn}} = P_{\text{turn}} r \psi_{\text{turn}}$, where ψ_{turn} is the angle of the turn, and

$$E_{\text{straight}} = E_{\text{start}} + E_{\text{stop}} + E_{\text{cruise}} \tag{35}$$
$$= P_{\text{start}} d_{\text{start}} + P_{\text{stop}} d_{\text{stop}} + P_{\text{cruise}} d_{\text{cruise}}, \tag{36}$$

where d_{stop} and d_{start} represent the respective straightaway distance component. To compute P_{cruise}, we use the same nonlinear program that was used to compute a minimum power turn (Equation 32), but set $r = \infty$ and use lift and drag coefficients for the low angle-of-attack regime. Approximating the stopping and starting energy is more complicated. This can be done by representing the stopping and starting maneuvers as a one dimensional ordinary differential equation where only the longitudinal velocity changes with time. To compute both acceleration energy and distance, we must find the velocity profile given as

$$v(t) = \int_{v_i}^{v_f} \dot{v} dt, \tag{37}$$

where v_i and v_f are the previously calculated cruise or turn velocities. For instance, when heading into a turn, the cruise velocity will be v_i and the turn velocity will be v_f. Assuming there is no acceleration in the vertical direction, both the quadcopter and the fixed-wing's longitudinal dynamics can be described as

$$\dot{v} = \frac{g}{\tan\theta} - c\rho v^2 \frac{S}{m}, \tag{38}$$

where $c = 0.5$ for the quadcopter and $c = 1$ for the fixed wing. For the quadcopter, we allow maximum throttle for acceleration and deceleration. Therefore

$$\theta = \pm \arcsin\left(\frac{mg}{f_{t,\max}}\right), \tag{39}$$

with the sign dependent on acceleration direction. For the fixed wing, $\theta(t)$ is the maximum value that satisfies the condition

$$f_t - \frac{mg}{\sin\theta} - \rho v^2 S \cos\theta = 0, \tag{40}$$

where $f_t \in [0, f_{t,\max}]$. Given $v(t)$, θ, and f_t, P_{start} and P_{stop} can be calculated from Equation 33. d_{start} and d_{stop} can be computed from an integration of $v(t)$. We can now compute the energy ratio $E_r = E_{fw}/E_{\text{quad}}$ for a given fixed wing and quadcopter pair for a flight path comprised of straightaways and turns. To gain some intuition about how energy ratio varies with vehicle scale and environment, we analyze the energy advantage of four pairs of vehicles (see Table 1) executing a flight path consisting of a straightaway and a 90° turn (see Figure 35). While we recognize that our analysis makes many assumptions and approximations, we still believe that it provides some valuable insights. First, the energy advantage at smaller vehicle scales is significantly greater than at larger scales. Secondly, fixed-wing UAVs almost always outperform quadrotors, except in very persistent hover conditions (i.e., when a flight path is entirely made up of tight turns), and often significantly outperform quadrotors (by a factor of two or more) at urban scales (e.g., Edge540XL with 20 m straightaways and 2 m turns). We also observed that accounting for turn maneuver starting and stopping energy did not dramatically impact the results except at very low turn radii.

Table 1. Parameters used for simplified dynamics model use to compute turn radii and energy.

	Vehicle	Mass (kg)	Surface Area (m^2)	Propeller Diameter (m)
(a)	Mini Edge 540	0.100	0.151	0.152
	Tello	0.100	0.022	0.0762
(b)	EdgeV3	0.450	0.369	0.279
	Mavic Air	0.450	0.022	0.135
(c)	Edge540XL	1.00	0.449	0.305
	Mavic2	1.00	0.022	0.220
(d)	Gamebird	3.00	0.670	0.406
	Matrice	3.00	0.022	0.330

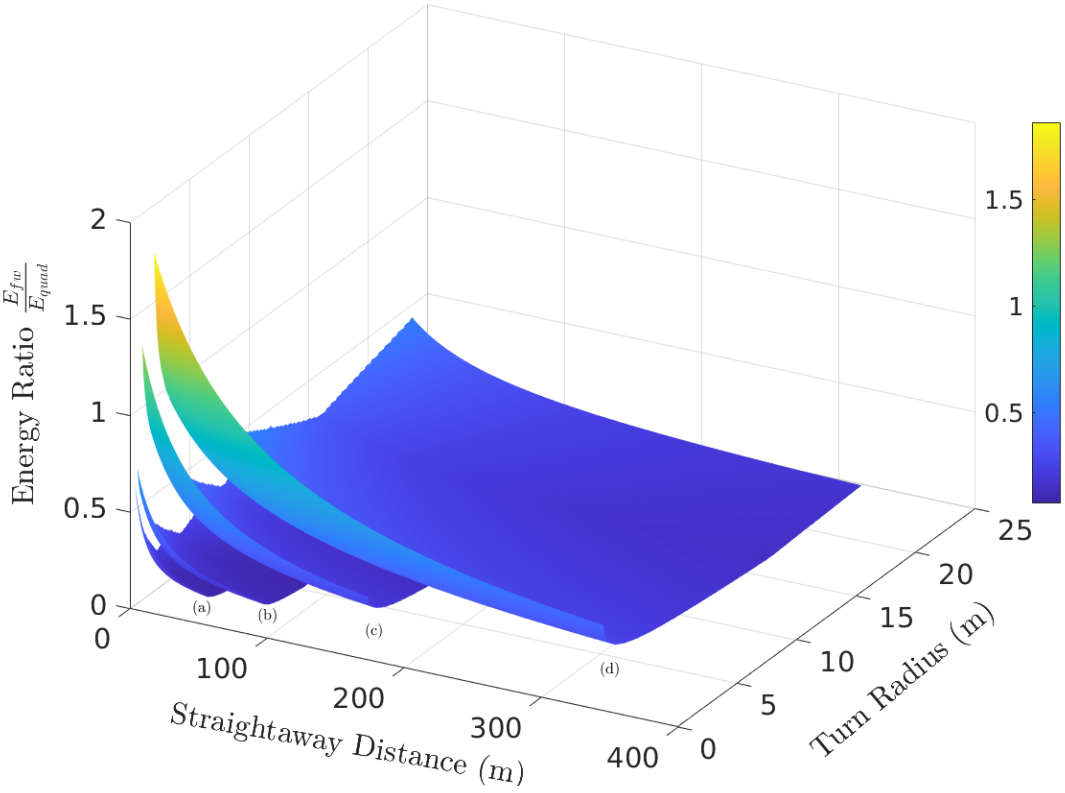


Figure 35. Plots of the energy ratio E_r of fixed-wing over quadcopter energy as a function of straightaway distance and radius of curvature. Plots (a)–(d) represent different energy ratios across pairs of vehicles with the same mass (see Table 1). The domain of plots is determined by the "maximum" turning radius (R_{max}) computed using the cruise trim condition. The maximum of the y domain corresponds to the maximum turning radius, while the maximum x domain is bounded by $10\frac{\pi}{2}R_{\text{max}}$.

ORCID

Max Basescu https://orcid.org/0009-0005-8582-5240
Adam Polevoy https://orcid.org/0000-0001-5201-7056
Bryanna Yeh https://orcid.org/0009-0002-7136-1340
Luca Scheuer https://orcid.org/0009-0008-1509-6072
Erin Sutton https://orcid.org/0000-0003-3586-2383
Joseph L. Moore https://orcid.org/0000-0003-4461-1666

References

Altug, E., Ostrowski, J. P., and Mahony, R. (2002). Control of a quadrotor helicopter using visual feedback. In *Proceedings 2002 IEEE international conference on robotics and automation (Cat. No. 02CH37292)*, volume 1, pages 72–77. IEEE.

Alturbeh, H. and Whidborne, J. F. (2014). Real-time obstacle collision avoidance for fixed wing aircraft using b-splines. In *Control (CONTROL), 2014 UKACC International Conference on*, pages 115–120. IEEE.

Arul, S. H., Sathyamoorthy, A. J., Patel, S., Otte, M., Xu, H., Lin, M. C., and Manocha, D. (2019). Lswarm: Efficient collision avoidance for large swarms with coverage constraints in complex urban scenes. *IEEE Robotics and Automation Letters*, 4(4):3940–3947.

Bachrach, A., Prentice, S., He, R., and Roy, N. (2011). Range–robust autonomous navigation in gps-denied environments. *Journal of Field Robotics*, 28(5):644–666.

Bangura, M. and Mahony, R. (2017). Thrust control for multirotor aerial vehicles. *IEEE Transactions on Robotics*, 33(2):390–405.

Barry, A. J., Florence, P. R., and Tedrake, R. (2018). High-speed autonomous obstacle avoidance with pushbroom stereo. *Journal of Field Robotics*, 35(1):52–68.

Basescu, M. and Moore, J. (2020). Direct nmpc for post-stall motion planning with fixed-wing UAVs. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 9592–9598.

Blösch, M., Weiss, S., Scaramuzza, D., and Siegwart, R. (2010). Vision based mav navigation in unknown and unstructured environments. In *2010 IEEE International Conference on Robotics and Automation*, pages 21–28. IEEE.

Bonzanini, A. D., Santos, T. L., and Mesbah, A. (2019). Tube-based stochastic nonlinear model predictive control: A comparative study on constraint tightening. *IFAC-PapersOnLine*, 52(1):598–603.

Bry, A., Richter, C., Bachrach, A., and Roy, N. (2015). Aggressive flight of fixed-wing and quadrotor aircraft in dense indoor environments. *The International Journal of Robotics Research*, 34(7):969–1002.

Bulka, E. and Nahon, M. (2018). Autonomous fixed-wing aerobatics: from theory to flight. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6573–6580. IEEE.

Bulka, E. and Nahon, M. (2019). High-speed obstacle-avoidance with agile fixed-wing aircraft. In *2019 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 971–980. IEEE.

Chauhan, S. S. (2020). *Optimization Studies for Aircraft Considering Propeller-Wing Interaction*. PhD thesis.

Chauhan, S. S. and Martins, J. R. (2020). Tilt-wing evtol takeoff trajectory optimization. *Journal of aircraft*, 57(1):93–112.

Chung, S.-J., Paranjape, A. A., Dames, P., Shen, S., and Kumar, V. (2018). A survey on aerial swarm robotics. *IEEE Transactions on Robotics*, 34(4):837–855.

Chung, T. H. (2021). Offensive swarm-enabled tactics (offset). DARPA.

Chung, T. H., Clement, M. R., Day, M. A., Jones, K. D., Davis, D., and Jones, M. (2016). Live-fly, large-scale field experimentation for large numbers of fixed-wing uavs. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1255–1262. IEEE.

Clark, S., Usbeck, K., Diller, D., and Schantz, R. E. (2021). Ccast: A framework and practical deployment of heterogeneous unmanned system swarms. *GetMobile: Mobile Computing and Communications*, 24(4): 17–26.

Cory, R. and Tedrake, R. (2008). Experiments in fixed-wing uav perching. In *AIAA Guidance, Navigation and Control Conference and Exhibit*.

Deits, R. and Tedrake, R. (2015). Efficient mixed-integer planning for uavs in cluttered environments. In *2015 IEEE international conference on robotics and automation (ICRA)*, pages 42–49. IEEE.

Dekka, A., Wu, B., and Zargari, N. R. (2016). An improved indirect model predictive control approach for modular multilevel converter. In *IECON 2016-42nd Annual Conference of the IEEE Industrial Electronics Society*, pages 5959–5964. IEEE.

Faessler, M., Fontana, F., Forster, C., Mueggler, E., Pizzoli, M., and Scaramuzza, D. (2016). Autonomous, vision-based flight and live dense 3d mapping with a quadrotor micro aerial vehicle. *Journal of Field Robotics*, 33(4):431–450.

Faessler, M., Franchi, A., and Scaramuzza, D. (2018). Differential flatness of quadrotor dynamics subject to rotor drag for accurate tracking of high-speed trajectories. *IEEE Robot. Autom. Lett*, 3(2):620–626.

Falanga, D., Foehn, P., Lu, P., and Scaramuzza, D. (2018). Pampc: Perception-aware model predictive control for quadrotors. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1–8. IEEE.

Florence, P., Carter, J., and Tedrake, R. (2020). Integrated perception and control at high speed: Evaluating collision avoidance maneuvers without maps. In *Algorithmic Foundations of Robotics XII*, pages 304–319. Springer.

Florence, P. R., Carter, J., Ware, J., and Tedrake, R. (2018). Nanomap: Fast, uncertainty-aware proximity queries with lazy search over local 3d data. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7631–7638. IEEE.

Gao, F., Wu, W., Gao, W., and Shen, S. (2019). Flying on point clouds: Online trajectory generation and autonomous navigation for quadrotors in cluttered environments. *Journal of Field Robotics*, 36(4):710–733.

Garimella, G., Sheckells, M., Moore, J. L., and Kobilarov, M. (2018). Robust obstacle avoidance using tube nmpc. In *2018 Robotics: Science and Systems (RSS) Conference*.

Gill, P., Murray, W., and Saunders, M. (2005). Snopt: An sqp algorithm for large-scale constrained optimization. *SIAM review*, 47(1):99–131.

Glauert, H. (1927). The theory of the autogyro. *The Aeronautical Journal*, 31(198):483–508.

Grebe, C., Wise, E., and Kelly, J. (2021). Observability-aware trajectory optimization: Theory, viability, and state of the art. In *2021 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI)*, pages 1–8. IEEE.

Grzonka, S., Grisetti, G., and Burgard, W. (2009). Towards a navigation system for autonomous indoor flying. In *2009 IEEE international conference on Robotics and Automation*, pages 2878–2883. IEEE.

Hamer, M., Widmer, L., and D'andrea, R. (2018). Fast generation of collision-free trajectories for robot swarms using gpu acceleration. *IEEE Access*, 7:6679–6690.

Hauser, J. and Hindman, R. (1997). Aggressive flight maneuvers. In *Proceedings of the 36th IEEE Conference on Decision and Control*, volume 5, pages 4186–4191 vol.5.

Herbst, W. B. (1984). Supermaneuverability. Technical report.

Hernández Ramírez, J. C. and Nahon, M. (2020). Nonlinear vector-projection control for agile fixed-wing unmanned aerial vehicles. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5314–5320.

Hoerner, S. F. and Borst, H. V. (1985). Fluid-dynamic lift: practical information on aerodynamic and hydrodynamic lift.

Hoffmann, G., Huang, H., Waslander, S., and Tomlin, C. (2007). Quadrotor helicopter flight dynamics and control: Theory and experiment. In *AIAA guidance, navigation and control conference and exhibit*, page 6461.

Hoffmann, G. M., Huang, H., Waslander, S. L., and Tomlin, C. J. (2011). Precision flight control for a multi-vehicle quadrotor helicopter testbed. *Control engineering practice*, 19(9):1023–1036.

Hornung, A., Wurm, K. M., Bennewitz, M., Stachniss, C., and Burgard, W. (2013). Octomap: An efficient probabilistic 3d mapping framework based on octrees. *Autonomous Robots*, 34(3):189–206.

Huang, A. S., Bachrach, A., Henry, P., Krainin, M., Maturana, D., Fox, D., and Roy, N. (2017). Visual odometry and mapping for autonomous flight using an rgb-d camera. In *Robotics Research*, pages 235–252. Springer.

Huang, H., Hoffmann, G. M., Waslander, S. L., and Tomlin, C. J. (2009). Aerodynamics and control of autonomous quadrotor helicopters in aggressive maneuvering. In *2009 IEEE international conference on robotics and automation*, pages 3277–3282. IEEE.

Johansen, T. A. (2011). Introduction to nonlinear model predictive control and moving horizon estimation. *Selected topics on constrained and nonlinear control*, 1:1–53.

Katyal, K. D., Polevoy, A., Moore, J., Knuth, C., and Popek, K. M. (2021). High-speed robot navigation using predicted occupancy maps. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5476–5482. IEEE.

Keller, J., Thakur, D., Likhachev, M., Gallier, J., and Kumar, V. (2016). Coordinated path planning for fixed-wing uas conducting persistent surveillance missions. *IEEE Transactions on Automation Science and Engineering*, 14(1):17–24.

Khan, W. and Nahon, M. (2016). Modeling dynamics of agile fixed-wing uavs for real-time applications. In *2016 international conference on unmanned aircraft systems (ICUAS)*, pages 1303–1312. IEEE.

Lavalle, S. (1998). Rapidly-exploring random trees: A new tool for path planning. Technical report.

Levin, J. M., Nahon, M., and Paranjape, A. A. (2016). Aggressive turn-around manoeuvres with an agile fixed-wing uav. *IFAC-PapersOnLine*, 49(17):242–247.

Levin, J. M., Nahon, M., and Paranjape, A. A. (2019). Real-time motion planning with a fixed-wing uav using an agile maneuver space. *Autonomous Robots*, pages 1–20.

Levin, J. M., Paranjape, A., and Nahon, M. (2017). Agile fixed-wing uav motion planning with knife-edge maneuvers. In *2017 International Conference on Unmanned Aircraft Systems*, pages 114–123. IEEE.

Liew, C. F., DeLatte, D., Takeishi, N., and Yairi, T. (2017). Recent developments in aerial robotics: An survey and prototypes overview. *arXiv preprint arXiv:1711.10085*.

Majumdar, A. and Tedrake, R. (2017). Funnel libraries for real-time robust feedback motion planning. *The International Journal of Robotics Research*, 36(8):947–982.

Manchester, Z. and Kuindersma, S. (2017). Dirtrel: Robust trajectory optimization with ellipsoidal disturbances and lqr feedback. In *Robotics: Science and Systems*.

Mathisen, S., Gryte, K., Gros, S., and Johansen, T. A. (2020). Precision deep-stall landing of fixed-wing uavs using nonlinear model predictive control. *Journal of Intelligent and Robotics Systems*, 101(24).

Mathisen, S., Gryte, K., Gros, S., and Johansen, T. A. (2021). Precision deep-stall landing of fixed-wing uavs using nonlinear model predictive control. *Journal of Intelligent & Robotic Systems*, 101(1):1–15.

Mathisen, S. H., Gryte, K., Johansen, T., and Fossen, T. I. (2016). Non-linear model predictive control for longitudinal and lateral guidance of a small fixed-wing uav in precision deep stall landing. In *AIAA Infotech @ Aerospace*, page 0512. AIAA.

Matsumoto, T., Konno, A., Suzuki, R., Oosedo, A., Go, K., and Uchiyama, M. (2010). Agile turnaround using post-stall maneuvers for tail-sitter vtol uavs. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1612–1617. IEEE.

McGrew, J. S., How, J. P., Williams, B., and Roy, N. (2010). Air-combat strategy using approximate dynamic programming. *Journal of guidance, control, and dynamics*, 33(5):1641–1654.

Mellinger, D. and Kumar, V. (2011). Minimum snap trajectory generation and control for quadrotors. In *2011 IEEE International Conference on Robotics and Automation*, pages 2520–2525. IEEE.

Mellinger, D., Michael, N., and Kumar, V. (2012). Trajectory generation and control for precise aggressive maneuvers with quadrotors. *The International Journal of Robotics Research*, 31(5):664–674.

Milam, M. B., Franz, R., and Murray, R. M. (2002). Real-time constrained trajectory generation applied to a flight control experiment. *IFAC Proceedings Volumes*, 35(1):175–180.

Milam, M. B., Mushambi, K., and Murray, R. M. (2000). A new computational approach to real-time trajectory generation for constrained mechanical systems. In *Decision and Control, 2000. Proceedings of the 39th IEEE Conference on*, volume 1, pages 845–851. IEEE.

Min, P. (2004 - 2019). binvox. http://www.patrickmin.com/binvox or https://www.google.com/search?q=binvox. Accessed: 2020-12-31.

Moore, J., Cory, R., and Tedrake, R. (2014). Robust post-stall perching with a simple fixed-wing glider using LQR-trees. *Bioinspiration & Biomimetics*, 9(2).

Moore, J., Fein, A., and Setzler, W. (2018). Design and analysis of a fixed-wing unmanned aerial-aquatic vehicle. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1236–1243. IEEE.

Murray, R. M. (2007). Recent research in cooperative control of multivehicle systems.

Naderi, K., Rajamäki, J., and Hämäläinen, P. (2015). Rt-rrt* a real-time path planning algorithm based on rrt. In *Proceedings of the 8th ACM SIGGRAPH Conference on Motion in Games*, pages 113–118.

Neunert, M., De Crousaz, C., Furrer, F., Kamel, M., Farshidian, F., Siegwart, R., and Buchli, J. (2016). Fast nonlinear model predictive control for unified trajectory optimization and tracking. In *2016 IEEE international conference on robotics and automation (ICRA)*, pages 1398–1404. IEEE.

Nooruddin, F. S. and Turk, G. (2003). Simplification and repair of polygonal models using volumetric techniques. *IEEE Transactions on Visualization and Computer Graphics*, 9(2):191–205.

Pardo, D., Möller, L., Neunert, M., Winkler, A. W., and Buchli, J. (2016). Evaluating direct transcription and nonlinear optimization methods for robot motion planning. *IEEE Robotics and Automation Letters*, 1(2):946–953.

Park, S. (2020). Control and guidance for precision deep stall landing. *Journal of Guidance, Control, and Dynamics*, 43(2):365–372.

Park, S., Deyst, J., and How, J. (2004). A new nonlinear guidance logic for trajectory tracking. In *AIAA Guidance, Navigation, and Control Conference and Exhibit*.

Peloquin, R.-A., Thibault, D., and Desbiens, A. L. (2016). Design of a passive vertical takeoff and landing aquatic uav. *IEEE Robotics and Automation Letters*, 2(2):381–388.

Polevoy, A., Basescu, M., Scheuer, L., and Moore, J. (2022). Post-stall navigation with fixed-wing uavs using onboard vision. In *2022 IEEE International Conference on Robotics and Automation (ICRA)*.

Preiss, J. A., Honig, W., Sukhatme, G. S., and Ayanian, N. (2017). Crazyswarm: A large nano-quadcopter swarm. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3299–3304. IEEE.

Rehan, M., Akram, F., Shahzad, A., Shams, T., and Ali, Q. (2022). Vertical take-off and landing hybrid unmanned aerial vehicles: An overview. *The Aeronautical Journal*, pages 1–41.

Richter, C., Bry, A., and Roy, N. (2016). Polynomial trajectory planning for aggressive quadrotor flight in dense indoor environments. In *Robotics Research*, pages 649–666. Springer.

Selig, M. S. (2014). Real-time flight simulation of highly maneuverable unmanned aerial vehicles. *Journal of Aircraft*, 51(6):1705–1725.

Shen, S., Michael, N., and Kumar, V. (2011). Autonomous multi-floor indoor navigation with a computationally constrained mav. In *2011 IEEE International Conference on Robotics and Automation*, pages 20–25. IEEE.

Singh, S., Landry, B., Majumdar, A., Slotine, J.-J., and Pavone, M. (2019). Robust feedback motion planning via contraction theory. *The International Journal of Robotics Research*.

Sobolic, F. and How, J. (2009). Nonlinear agile control test bed for a fixed wing aircraft in a constrained environment. In *AIAA Infotech@ Aerospace Conference and AIAA Unmanned... Unlimited Conference*, page 1927.

Stastny, T. and Siegwart, R. (2018). Nonlinear model predictive guidance for fixed-wing uavs using identified control augmented dynamics. In *2018 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 432–442. IEEE.

Strickland, L., Day, M. A., DeMarco, K., Squires, E., and Pippin, C. (2018). Responding to unmanned aerial swarm saturation attacks with autonomous counter-swarms. In *Ground/Air Multisensor Interoperability, Integration, and Networking for Persistent ISR IX*, volume 10635, pages 247–263. SPIE.

Tabib, W. and Michael, N. (2021). Simultaneous localization and mapping of subterranean voids with gaussian mixture models. In *Field and Service Robotics*, pages 173–187. Springer.

Team, A. et al. (2022a). Ardupilot. *URL: www. ardupilot. org, accessed*, 2:12.

Team, A. et al. (2022b). Ardupilot. *URL: https://ardupilot.org/dev/docs/ekf2-estimation-system.html, accessed*, 2:12.

Tedrake, R. (2009). Underactuated robotics: Learning, planning, and control for efficient and agile machines: Course notes for mit 6.832. *Working draft edition*, page 3.

Turpin, M., Michael, N., and Kumar, V. (2014). Capt: Concurrent assignment and planning of trajectories for multiple robots. *The International Journal of Robotics Research*, 33(1):98–112.

Umari, H. and Mukhopadhyay, S. (2017). Autonomous robotic exploration based on multiple rapidly-exploring randomized trees. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1396–1402. IEEE.

Wang, J. and Olson, E. (2016). AprilTag 2: Efficient and robust fiducial detection. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4193–4198. IEEE.

Waslander, S. L., Hoffmann, G. M., Jang, J. S., and Tomlin, C. J. (2005). Multi-agent quadrotor testbed control design: Integral sliding mode vs. reinforcement learning. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2–6. Citeseer.

Yang, K. and Sukkarieh, S. (2008). 3d smooth path planning for a uav in cluttered natural environments. In *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 794–800. IEEE.

Yang, K. and Sukkarieh, S. (2010). An analytical continuous-curvature path-smoothing algorithm. *IEEE Transactions on Robotics*, 26(3):561–568.

Zhou, X., Zhu, J., Zhou, H., Xu, C., and Gao, F. (2021). Ego-swarm: A fully autonomous and decentralized quadrotor swarm system in cluttered environments. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4101–4107. IEEE.

How to cite this article: Basescu, M., Polevoy, A., Yeh, B., Scheuer, L., Sutton, E., & Moore, J. L. (2023). Agile fixed-wing UAVs for urban swarm operations. *Field Robotics, 3*, 725–765.