Visual Modeling System for Optimization-Based Real-Time Trajectory Planning for Autonomous Aerial Drones

Conference Paper · March 2022

DOI: 10.1109/AERO53065.2022.9843495

CITATIONS
3

READS
61

7 authors, including:

Skye Mceowen
University of Washington Seattle
10 PUBLICATIONS **53** CITATIONS
SEE PROFILE

Behçet Açıkmeşe
University of Washington Seattle
264 PUBLICATIONS **5,297** CITATIONS
SEE PROFILE

Visual Modeling System for Optimization-Based Real-Time Trajectory Planning for Autonomous Aerial Drones

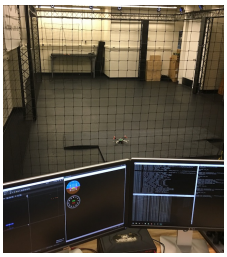
Skye Mceowen, Daniel Sullivan, Oliver Sheridan, Dan Calderone, and Behçet Açıkmeşe
Department of Aeronautics and Astronautics
University of Washington
Seattle, WA 98105, USA
skye95@uw.edu

Benjamin Chasnov
Department of Electrical and Computer Engineering
University of Washington
Seattle, WA 98105, USA
bchasnov@uw.edu

***Abstract*— In this paper, we present a visual modeling system to enable users to seamlessly describe the constraints of trajectory planning problems for autonomous aerial drones. The proposed modeling system comes with an intuitive GUI-based interface that enables the user to specify trajectory objectives, add and remove motion constraints, and update the constraint parameters in real-time. The interface algorithm acts as a high-level parser to convert graphically specified constraints into a standard form of the underlying optimal control problem. Then a sequence of convex optimization problems, convex subproblems, are generated whose solutions will converge to a solution of the trajectory planning problem. This convex optimization based method is referred to as *successive convexification* (SCvx) [1]. Beneath the interface, there is another low-level layer of problem parsing, which aims to model each convex subproblem as a Second Order Cone Programming (SOCP) problem in a standard form. Once each SOCP is formulated in this standard form, it can be passed to our in-house developed primal-dual interior point method (IPM) SOCP solver [2, 3] to obtain a solution for each convex subproblem within SCvx. This paper is aimed to describe the functional architecture of the visual modeling system and its core algorithms, and also presents some illustrative flight experiments.**

(a) Real-time visual modeling system for optimization-based trajectory planning of indoor and outdoor flights.

(b) Custom quadrotor.



(c) Motion capture and ground control system.

Figure 1: A user of the visual modeling system specifies relevant constraints on a physics-first problem, shown in (a). These specifications are translated into meaningful constraints on the optimization problem that generates dynamically feasible trajectories for the vehicle pictured in (b). The vehicle's trajectory can be updated mid-flight as mission specifications change, enabling real-time decision-making for highly dynamic scenarios. (c) Our lab setup allows for rapid prototyping and evaluation of control and planning algorithms, and has proven to be indispensable for system identification and tuning the parameters associated with the optimizers.

TABLE OF CONTENTS

1. INTRODUCTION 1
2. BACKGROUND 2
3. VISUAL MODELING SYSTEM 4
4. HIGH-LEVEL AND LOW-LEVEL PARSERS 6
5. EXPERIMENTAL DEMONSTRATIONS 6
6. CONCLUSION & FUTURE WORK 7
REFERENCES 8

1. INTRODUCTION

Optimization-based trajectory planning has become a crucial part of many autonomous aerospace vehicles [4–7]. Research and tools for optimization-based trajectory planning utilizing successive solutions to convex optimization subproblems in order to refine trajectories have found widespread use for aerospace applications [8–18]. Specifically, the framework of *successive convexification* (SCvx) has been developed as a real-time methodology that has efficient convergence properties [19,20]. This framework has proven to be effective in trajectory planning with applications to rocket landing and quadrotor trajectory planning. However, there is much room for improvement in the realm of user transparency. Historically, implementing SCvx requires an extensive amount of analytical work to be done a priori in order to formulate a nonlinear optimal control problem, which is subsequently reformulated as a sequence of convex subproblems. Before implementation, a discretization scheme must be determined and applied to set up the convex subproblems. This analytical process takes time, and requires an expert level of knowledge of the theory behind these methods.

However, real-world trajectory planning scenarios often change rapidly, requiring modification of trajectories in real-

978-1-6654-3760-8/22/$31.00 ©2022 IEEE

time. Additionally, some scenarios are too ambiguous to allow for a priori planning, which indicates that the process of manually reformulating the problem can become inhibiting to such complex, real-time scenarios. A trajectory planning framework that gives the user a clear picture of the optimal solutions as well as easy access to modify the problem parameters and recompute solutions would be a leap forward in the practical use of optimization-based algorithms to aerospace and robotic trajectory planning problems.

Graphical user interfaces (GUI's) and natural user interfaces (NUI's) have been used for robotic planning in a variety of contexts from drone control [21, 22] to manipulator control [23]. Drone applications in general reduce user inputs to waypoints and times and choose trajectories to optimize some objective such as minimum time or minimum snap. Along with graphical user interfaces, many techniques have focused on controlling robots through speech, gestures, body position, and haptic feedback in virtual or augmented reality environment [24–27]. Applications include industrial trajectory planning [28, 29] as well as performance [30] and videography [31, 32]. Some applications allow for communication of intention from the robot to the human as well [33, 34]. Higher level controllers have also been used to break down robot tasks into simple building blocks that a user can choose from [35–37].

In this paper, we present a touchscreen tablet based visual modeling system that enables intuitive human interaction in formulating trajectory planning (optimal control) problems for aerial drones in real-time. The graphical interface allows the user to quickly configure nonconvex constraints for trajectory planning problems. For example, constraints may include elliptical keep-out zones or polyhedrons, which can either be placed by the user via the touch screen or anchored to a beacon being tracked in the environment. This visual modeling system then acts as a high-level parser, converting the graphically-defined constraints placed on a map of the flight space into a convex subproblem standard form that can be passed to the SCvx algorithm software to generate a trajectory satisfying these constraints. A sequence of these convex subproblems are generated recursively and solved numerically until convergence. Further details are provided in Section 2. Beneath the visual interface lies a low-level parser that models each convex subproblem as a Second Order Cone Programming (SOCP) problem in a standard form, as in Section 4. This parsing sequence is shown in Fig. 2. The visual interface augments the user's ability to make high-level decisions, while relying on i) SCvx's strong theoretical convergence properties and ii) efficiency of the numerical solution algorithms for convex optimization to ensure trajectory solutions in real-time (Fig. 3).

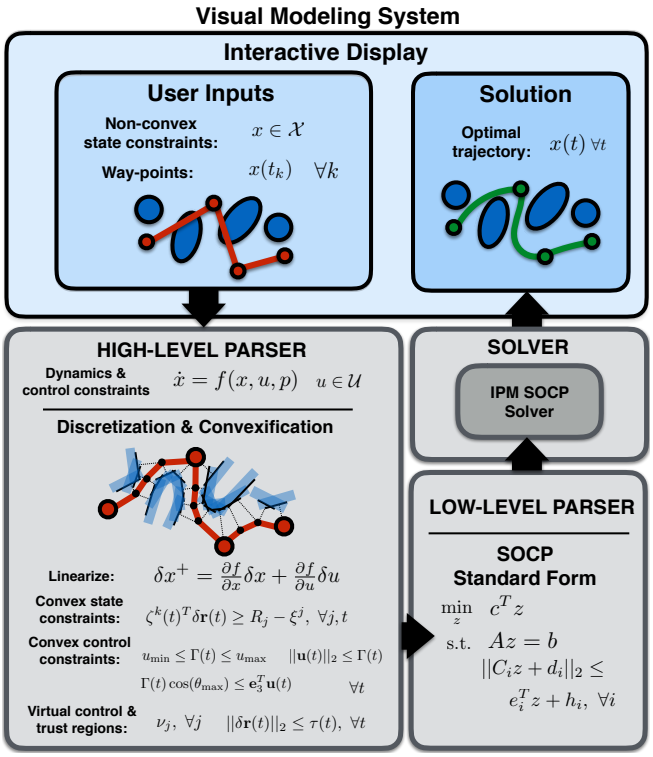


Figure 2: The layers of parsing to convert the user-defined graphical constraints in the visual optimization interface into a form that can be solved via the SCvx algorithm to produce dynamically feasible solutions are depicted. The Visual Modeling System Interactive Display and High-Level Parser are the novel implementation contributions presented in this paper.

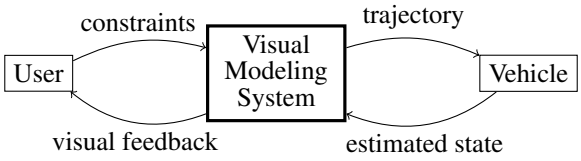


Figure 3: The Visual Modeling System allows for the negotiation of constraints with the user, while adhering to the constraints of the physical vehicle. The user is in a closed-loop system with the vehicle. The state of the vehicle is used as the initial boundary condition of the optimal control problem, and the user receives feedback on the state and progress of the mission objectives.

2. BACKGROUND

To demonstrate live tracking of our real-time trajectory generation techniques, we employ our research group's custom aerial drone hardware and software test bed. These drones track the trajectories as they are commanded by a user via the visual modeling system interface over a WiFi network. These key components and how they interact are discussed in the sections below, with the visual modeling system running on a handheld tablet serving as a keystone between them.

GNC Hardware and Software Architecture

Here we give a brief overview of the key building blocks of the Guidance, Navigation and Control (GNC) software and hardware infrastructure used to generate experimental results in the Autonomous Controls Lab (ACL). The software and hardware are broken into three distinct packages for vehicle trajectory generation (guidance), vehicle state estimation (navigation), and vehicle trajectory tracking (control) which run on separate computers interfaced together via a WiFi network. The control is performed on-board ACL's custom-built quadrotor platforms which run a flight software (FSW) package written in-house in C++ on an Intel Edison SoC clocked at 500MHz [38]. The vehicles employ high-order feedback controllers that are outside of the scope of this paper. To perform closed-loop tracking on-board the vehicle, this FSW package ingests both the guidance trajectories and vehicle navigation data over the WiFi network as they are generated in real-time off-board. The indoor navigation is performed using an OptiTrack motion-capture system which supplies both position and attitude data at up to 180Hz with a precision down to 4mm, by employing Prime17W and Prime41 cameras. A custom navigation software package also written in C++ takes estimated vehicle state data from

this camera system and sends it to both the vehicle FSW for on-board closed-loop control as mentioned above, and to the SCvx software package running on the visual modeling system to generate real-time trajectories for the vehicle. For outdoor applications, the navigation is conducted on-board the vehicle's flight computer using a GPS and magnetometer, and an extended Kalman filter (EKF) for state estimation. Figure 1 displays the custom quadrotor drone, our indoor motion capture system, and a flight test at the outdoor flight facility. Further details about the control systems and state estimation are discussed in prior publications [38].

At the highest level, the trajectory generation is performed in real-time with a novel visual modeling system software package that combines the optimization-based technique of successive convexification (SCvx) developed in the ACL [1] with a graphical user interface (GUI) to allow streamlined human-algorithm interaction on a Microsoft Surface tablet.

Successive Convexification Overview: Trajectory Generation Guidance Algorithm

In this section, we summarize the mathematical formulation of ACL's SCvx optimization-based guidance used by the visual modeling system to generate trajectories in real-time that are linked to the aerial drone's initial state.

Generalized Non-Convex Formulation—We consider optimal control problems of the form:

$$\min_{u,p} \; J(x,u,p) \tag{1a}$$

$$\text{s.t.} \;\; \dot{x}(t) = f(x(t),u(t),p), \tag{1b}$$

$$x(t) \in \mathcal{X}, u(t) \in \mathcal{U}, \tag{1c}$$

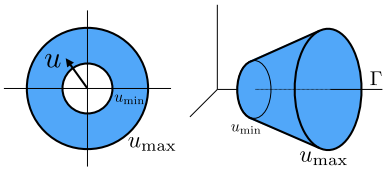
$$s(x(t),u(t),p) \leq 0, \tag{1d}$$

$$g(x(t),p) = 0 \tag{1e}$$

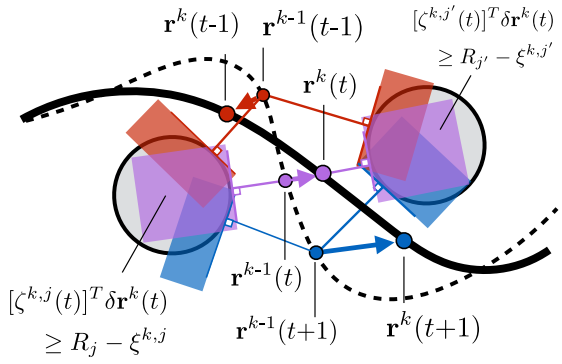
where $u(t)$ is a m-dimensional bounded control function, $x(t)$ is an n-dimensional state function, and $p \in \mathbb{R}^d$ is a vector of parameters. The function $f : \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^d \to \mathbb{R}^n$ is the vehicle dynamics, which is assumed to be at least once continuously differentiable. Initial and final boundary conditions are enforced by an affine constraints on $x(0)$ and $x(t_f)$. The convex state and control constraints are $\mathcal{X}$, $\mathcal{U}$ and the non-convex state and control constraints are captured by function $s : \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^d \to \mathbb{R}^{n_s}$, $g : \mathbb{R}^n \times \mathbb{R}^d \to \mathbb{R}^{n_g}$. The cost function J in the applications we study is typically minimum fuel or minimum time formulations, though in general any non-convex cost is supported. For the minimum time formulation, time is considered as a parameter.

Modeling and Convexification—To solve optimization problems using convex optimization, we must remove non-convexities and reframe problems as an SOCP problem [39]. The successive convexification framework involves several strategies for dealing with non-convexities in the optimal control formulation. Non-convex dynamics are linearized at each iteration $k = 1, 2, \ldots$ around the current trajectory.

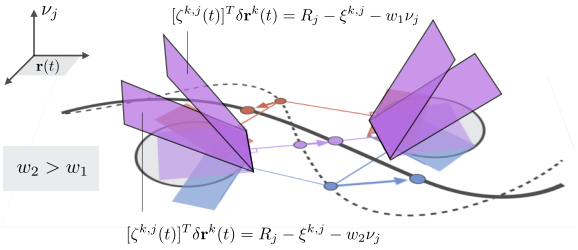
The real vehicle dynamics are six degree-of-freedom (6-DoF) dynamics corresponding to the three translational components and the three angular components of motion needed to describe the position and orientation in space at any given time. It is important to note that in reality the three translational degrees-of-freedom are coupled to the three angular degrees-of-freedom. However, we demonstrate that trajectories generated assuming simplified three degree-of-freedom



(a) Lossless convexification (control constraint).



(b) Obstacle avoidance (state constraint).



(c) Slack variables for feasibility

Figure 4: Sources of non-convexity in the constraints of the optimal control problem. (a) The non-convex minimum thrust condition (left) is convexified by introducing an additional variable to the optimization problem (right). (b) The non-convex ellipse keep-out zones are convexified by linearizing the constraint to form convex half-plane constraints for each sample point along the trajectory. (c) Illustration of how slack variables ν_j relax state constraints based on weight relative to the rest of the objective function.

(3-DoF) translational dynamics, neglecting the angular components, still produce dynamically feasible solutions that can be tracked with high-performance within a subset of the vehicle state space with respect to the orientation, rates, velocity and acceleration. This is done by imposing constraints on the state of the vehicle that the optimizer must adhere to when seeking a solution. For example, a state constraint might be enforced as a roll angle upper bound, or minimum and maximum acceleration values, written mathematically as an inequality constraint. In addition to dynamic and state constraints, control constraints limit upper and lower bounds of the control signal input used to actuate the vehicle along the trajectory. For the 3-DoF optimization problem, the state vector is given by $x(t) = (\mathbf{r}(t), \mathbf{v}(t))$ where $\mathbf{r}(t)$ and $\mathbf{v}(t)$ are the position and velocity of the quadrotor, respectively.

Control constraints with minimum thrust vector and minimum tilt angle bounds with the forms

$$0 < u_{\min} \leq ||\mathbf{u}(t)||_2 \leq u_{\max}, \quad ||\mathbf{u}(t)||_2 \cos(\theta_{\max}) \leq \mathbf{e}_3^T \mathbf{u}(t)$$

may be convexified by adding a slack variable $\Gamma(t)$

$$0 < u_{\min} \leq \Gamma(t) \leq u_{\max}, \;\; ||\mathbf{u}(t)||_2 \leq \Gamma(t),$$
$$\Gamma(t)\cos(\theta_{\max}) \leq \mathbf{e}_3^T \mathbf{u}(t)$$

without affecting the optimal solution. This *lossless convexification* is detailed in [38, 40].

Environmental constraints, represented by ellipsoidal keep-out zones, are described by the nonconvex constraints,

$$s_j = R_j - ||H_j(\mathbf{r}(t) - \mathbf{p}_j)||_2 \leq 0. \; \forall j \in \mathbb{J}$$

where $H_j = H_j^T \succeq 0$ determines the shape of the obstacle, and $R_j \geq 0$ determines the size, and $\mathbb{J}$ is a finite set of obstacles. Unlike the control constraints, the convexification of the environmental constraints must be re-computed at each optimization iteration as the trajectory is adjusted. For each time step t along the trajectory, each obstacle constraint is approximated by a half-space constraint defined by the tangent plane of the obstacle. At each iteration k, the constraints for obstacle j and time step t are given by

$$\xi^{k,j} + \left[\zeta^{k,j}(t)\right]^T \delta\mathbf{r}^k(t) \geq R_j - \nu_j, \;\; \nu_j \geq 0,$$

where

$$\Delta\mathbf{r}^{k,j}(t) \triangleq \mathbf{r}^{k-1}(t) - \mathbf{p}_j, \;\; \delta\mathbf{r}^k(t) \triangleq \mathbf{r}^k(t) - \mathbf{r}^{k-1}(t)$$
$$\xi^{k,j}(t) \triangleq \|H_j \Delta\mathbf{r}^{k,j}(t)\|_2, \;\; \zeta^{k,j}(t) \triangleq \frac{H_j^T H_j \Delta\mathbf{r}^{k,j}(t)}{\|H_j \Delta\mathbf{r}^{k,j}(t)\|_2}$$

Here $\mathbf{r}^k(t)$ is the trajectory point at time t, $\mathbf{r}^{k-1}(t)$ is that point at the previous optimization iteration, $\Delta\mathbf{r}^{k,j}(t)$ is the distance to the center of the j-th obstacle; $\delta\mathbf{r}^k(t)$ is the update to the position of $\mathbf{r}^k(t)$. ν_j is a slack variable that allows the constraints to be temporarily violated during trajectory planning. Each ν_j term is heavily penalized in the objective function forcing the final trajectory outside of each of the obstacles. Convexification of control lower bounds and successive convexification of obstacles is illustrated in Fig. 4.

More complicated constraint geometries or those that are time-dependent can be described using State-Triggered Constraints (STCs). An STC consists of a trigger condition and a corresponding constraint [41]. These are constraints that allow discrete logic to be integrated into a continuous optimization framework, that are switched on or off depending on the state of the vehicle. A common example is a hoop constraint, which can be represented as the trigger of the vehicle passing within an outer cylindrical corridor and the constraint where the vehicle is forced to fly within an inner cylindrical corridor. This can be generalized to any convex state-based trigger and corresponding constraint.

Artificial Unboundedness—Between iterations, the optimizer must make a discrete leap from one set of parameters and corresponding cost and another, described by $\delta\mathbf{r}^k(t)$. However, because each problem is obtained by linearizing the previous iteration, it is important that the optimizer does not make too large of a leap away from the region where the linearization is valid. To solve for this, we introduce trust regions

$$\|\delta\mathbf{r}^k(t)\|_2 \leq \tau(t), \; \forall t \in [0, t_f]$$

which penalize large steps between iterations by adding the penality term $\int_0^{t_f} \tau(t)^2$ to the objective function.

Aerial Drone Optimization Problem

The objective function consists of either minimum time or minimum fuel. Minimum fuel can either be constructed as fixed-final time, where the time of arrival at the terminal state constraint is fixed, or as free-final time where the time of arrival at the terminal state constraint is also an optimization variable. Minimum time is another way to formulate the problem, where flight duration is itself the sole cost being minimized.

For the first iteration, the problem must be initialized with a trajectory to begin optimizing from; this is typically chosen as a straight line, but can be chosen arbitrarily. At each subsequent iteration k, the following convex optimization problem is solved for the 3DoF quadrotor dynamics:

$$\underset{\mathbf{u}^k(t),\Gamma^k(t),\tau(t)}{\text{minimize}} \quad w\int_0^{t_f} \left(\Gamma^k(t)\right)^2 dt + \sum_{j\in J}\nu_j + \lambda\int_0^{t_f}\tau(t)^2$$

subject to:

$$\mathbf{r}^k(0) = \mathbf{r}_i, \qquad \mathbf{v}^k(0) = \mathbf{v}_0, \qquad \mathbf{u}^k(0) = g\mathbf{e}_3$$
$$\mathbf{r}^k(t_f) = \mathbf{r}_f, \qquad \mathbf{v}^k(t_f) = \mathbf{v}_f, \qquad \mathbf{u}^k(t_f) = g\mathbf{e}_3$$
$$\dot{\mathbf{r}}^k(t) = \mathbf{v}^k(t), \;\; \dot{\mathbf{v}}^k(t) = \mathbf{u}^k(t) - g\mathbf{e}_3$$

$$0 < u_{min} \leq \Gamma^k(t) \leq u_{max}, \;\; \|\mathbf{u}^k(t)\|_2 \leq \Gamma^k(t),$$
$$\Gamma^k(t)\cos(\theta_{max}) \leq \mathbf{e}_3^T\mathbf{u}^k(t), \;\; \|\delta\mathbf{r}^k(t)\|_2 \leq \tau(t)$$

For all $j \in \mathbb{J}$ and for $t \in [0, t_f]$:

$$\xi^{k,j} + \left[\zeta^{k,j}(t)\right]^T \delta\mathbf{r}^k(t) \geq R_j - \nu_j, \;\; \nu_j \geq 0$$

where $H_j \succeq 0$, $R_j \geq 0$ and

$$\Delta\mathbf{r}^{k,j}(t) \triangleq \mathbf{r}^{k-1}(t) - \mathbf{p}_j, \quad \delta\mathbf{r}^k(t) \triangleq \mathbf{r}^k(t) - \mathbf{r}^{k-1}(t),$$
$$\xi^{k,j}(t) \triangleq \|H_j\Delta\mathbf{r}^{k,j}(t)\|_2, \quad \zeta^{k,j}(t) \triangleq \frac{H_j^T H_j \Delta\mathbf{r}^{k,j}(t)}{\|H_j\Delta\mathbf{r}^{k,j}(t)\|_2}.$$

Successive convexification is not guaranteed to achieve convergence from any set of initializations, cost functions and constraints. The convergence properties vary with the many inputs, but we demonstrate this algorithm to run successfully in real-time as a quadrotor moves throughout a flight space. As the quadrotor moves fairly rapidly throughout the flight space, any rare and temporary nonconvergence issues due to local conditions are quickly replaced by a new feasible solution at the vehicle's next position [42, 43].

3. Visual Modeling System

We describe the functional architecture of the visual modeling system and its core algorithms, which enables intuitive human interaction in formulating trajectory planning (optimal control) problems for aerial drones in real-time. The interface algorithms convert graphically specified constraints into a standard form of the underlying optimal control problem.

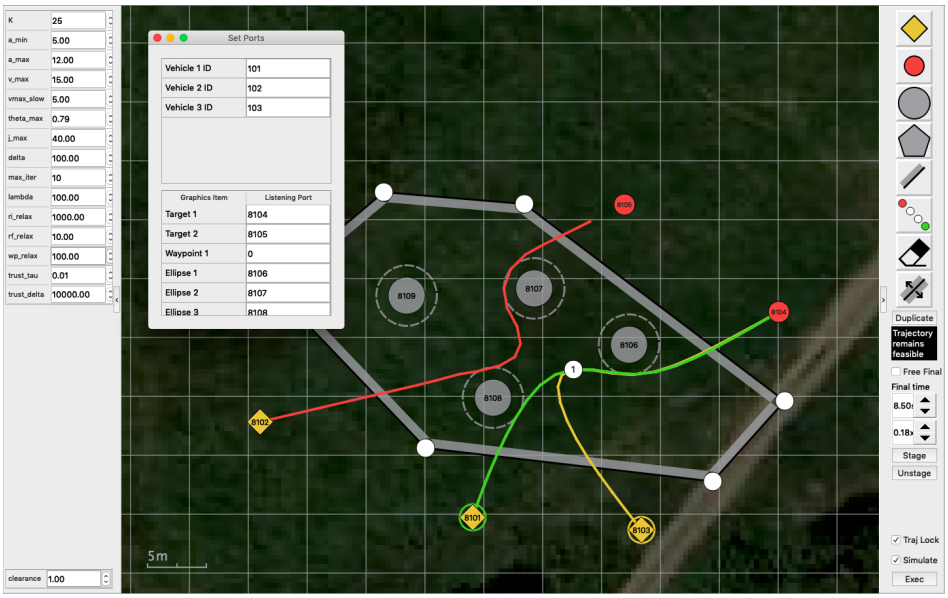


Figure 5: An overview of the visual autonomy interface. Yellow diamonds are the vehicles' current state, red circles are target destinations, and white numbered circles are waypoints. The gray circles are obstacles and the gray polytope is a slow zone. Each vehicle, target, and obstacle can be tagged to a live tracker using the pop-up window to be updated in real-time. The right panel provides an intuitive interface for non-expert users to update the constraints of the problem, whereas the left panel provides an detailed interface for expert users to fine-tune the optimization algorithms.

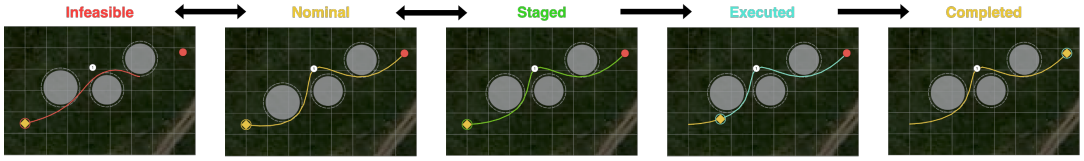


Figure 6: The colors of trajectories represent different stages of execution. Under nominal conditions (yellow), no constraints are violated and the goal can be reached. If the problem is infeasible (red), the users will be alerted on which parameters to adjust to regain feasibility. Once staged (green), the trajectory is locked and is readied for execution (blue). Once completed, it returns to the nominal stage.

User Inputs

The graphical interface allows the user to quickly configure nonconvex constraints for trajectory planning problems. For example, constraints may include elliptical keep-out zones or polyhedrons, which can either be placed on the map by the user via the touch screen or anchored to a beacon being tracked in the environment, as shown in Fig. 5. Problem parameters are updated both as the user defines constraints and as the autonomous drone moves throughout the environment, allowing relevant trajectories to be computed throughout the duration of the flight.

Boundary Conditions—The simplest use case for the tablet interface is defining initial and final conditions for a trajectory. Users can place a variety of target destinations on the map. One active target can be designated from a set of pre-placed destinations for the aerial drone. The drone's current telemetry is used as the initial condition.

State and Control Constraints— The interface also allows users to apply various constraints to the trajectory between the initial and final condition. The user can create constraints to avoid or keep-out zones by using an obstacle generation tool. The "obstacle create" mode allows users to place elliptical obstacles scaled according to the zoom level of the map. The user can then use the default mode to rotate, elongate, move, and duplicate obstacles by selecting the item and using the graphical handles that appear. Users can also place waypoints for the drone to pass through en route to the final destination along the trajectory. Waypoints are applied as hard positional constraints with no constraint on acceleration or velocity. Fig. 5 features four obstacles with dashed-line clearance zones and one waypoint, indicated by the numbered white dot.

State-Triggered Constraints—Finally, users can impose state-triggered constraints (STCs) which only trigger under certain conditions. In the visual modeling interface, for example, users can impose a "slow zone" on the map where the maximum velocity constraint is reduced. The slow zone region is defined via the polygon and hyperplane creation tools. The user either designates an entire halfspace region by placing two points with the hyperplane creation tool, or alternatively places a series of points defining the vertices of a polygon to specify an isolated intersection of halfspaces. These shapes can be moved or modified with graphical handles similar to the elliptical obstacles, and hyperplane direction can by flipped as well. Any vehicle within the intersection of these zones would then be subject to the lower maximum velocity constraint, which would then be passed to the low-level parser. Fig. 5 demonstrates a slow-zone defined by the boundaries of a user-created polygon.

Expert-User Optimization Parameters—While the high-level parser currently only supports a predefined set of convexified 3-DoF dynamics for aerial drones, the graphical interface provides options for expert users to fine tune various optimal control problem parameters. A cohesive description of these parameters is outside the scope of this paper. However,

examples include weighting values on the virtual control and trust region elements of the cost function, as well as the minimum and maximum bounds for vehicle acceleration.

Tablet Display

The visual interface reports a trajectory satisfying the constraints and parameters set by the user. This trajectory is displayed along with indicators about the feasibility of the solution. This overall status of constraints, obstacles, vehicles, targets, waypoints, trajectory, and feasibility are displayed to the user in an interactive format on the tablet screen. The user can then issue commands based on this state, such as uploading the trajectory to the vehicle if a feasible trajectory is reported. On the tablet screen, the trajectory status and vehicle position are continuously updated on the displayed map. The user is in a closed-loop system with the vehicle and optimizer (see Fig. 3), and can carefully operate and monitor the scenarios in real-time. The autonomous interface augments the user's ability to make high-level decisions, while relying on the optimizers strong theoretical guarantees.

Once the user selects a target for a vehicle, the GUI continuously runs SCvx at a rate of up to 10Hz to generate trajectories using all the problem inputs [38]. The status of the resulting trajectories is displayed using color codes and feedback messages. The progression of color codes is demonstrated in Fig. 6. Infeasible trajectories are displayed in red and the offending input is presented in a message box. Nominal trajectories are presented in yellow. Since these trajectories are continuously updated, the user must stage the trajectory of the currently selected drone to prevent it from updating further and validate it is the desired trajectory. The staged trajectory and associated vehicle are displayed in green. The user then executes the staged trajectory, either in simulation mode or transmitting it to the tagged vehicle. The executing trajectory and drone is displayed in blue. Depending on the application, the user may choose to lock the trajectory until complete or change to a newly computed trajectory on the fly.

4. High-Level and Low-Level Parsers

The visual modeling system acts as a high-level parser, converting the graphically-defined constraints placed on a map of the flight space into a convex subproblem standard form that can be passed to the SCvx algorithm software to generate a trajectory satisfying these constraints. A sequence of these convex subproblems, are generated whose solutions will converge to a solution of the trajectory planning problem. The optimal control problem implemented here for minimum-fuel or minimum time objectives for autonomous aerial drones is discussed in detail in Section 2. Beneath the interface is a low-level problem parser, which aims to model each convex subproblem as an SOCP in a standard form. This relation between the two layers of parsing is depicted in Fig. 2.

High-Level Parser: Convexification and Discretization

The high-level parser converts the nonconvex environmental constraints into their locally convex approximations. These convexified constraints are brought together with the vehicle dynamics, the originally convex constraints, and the cost function (assumed to be also convex) to form a standard convex subproblem. The problem is initialized with a straight line interpolation between the initial condition, waypoints, and final condition. A sequence of these convex subproblems is generated recursively and solved until convergence, where each problem is convexified locally around the previous solution. The converged final solution achieves a locally optimal solution to the original nonconvex optimal control problem.

Low-Level Parser and SOCP Solver

The low-level parser receives the convex subproblems from the high-level parser, which then reformulates these convex problems as a second-order cone programming (SOCP) problem in a standard form. Once each subproblem is expressed in a standard form, they are passed to our in-house developed primal-dual interior point method (IPM) solver to obtain a solution [2, 3]. This custom IPM solver runs very efficiently and therefore helps reduce overall algorithm run-time. Upon convergence, the solver passes the trajectory solution back to the visual modeling system for display to the user.

5. Experimental Demonstrations

Experimental demonstrations were performed and are intended as illustrative examples of the visual modeling interface in action. Video footage and commentary for each scenario presented in this section can be found on the Autonomous Controls Laboratory website at the following link:

https://depts.washington.edu/uwacl/media/aeroconf-2022-submission/

A snapshot is shown in Fig. 7, and the test scenarios are described below.

Scenario 1: Trajectory Planning with Boundary Conditions and Constraints

This scenario demonstrates simple trajectory with boundary conditions and constraints. The user executes multiple trajectories with a single vehicle and target in an obstacle course of multiple elliptical obstacles and user-defined waypoints.

Scenario 2: Live Trajectory Replanning

This scenario highlights the iterative properties of SCvx and shows the speed with which it can converge to a feasible solution. The user successfully executes new trajectories while the vehicle is mid-flight, seamlessly changing the course of the vehicle.

Scenario 3: Free Final Time Implementation

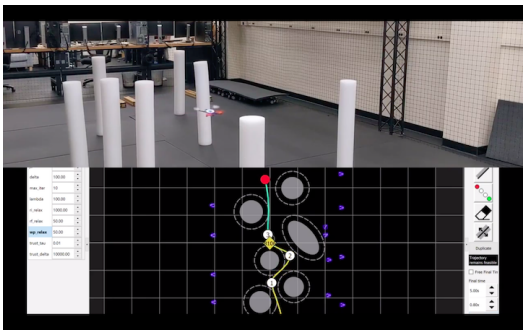
This scenario demonstrates a free final time implementation, where the final time is also an optimization variable instead of a user-defined boundary constraint on the terminal condition enforced on the vehicle. The user is able to modify the problem type and optimization variables through the tablet interface to successfully generate and execute minimum-time trajectories.

Scenario 4: Slow Zones via State-Triggered Constraints

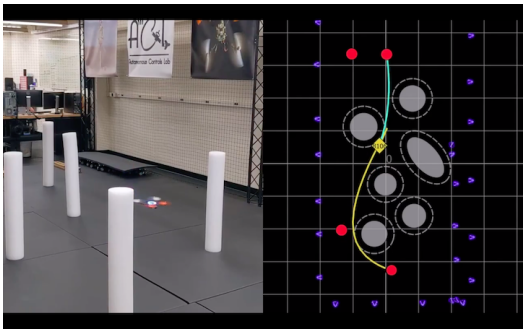
This example demonstrates state-triggered constraints (STCs) in the form of user defined slow zones. The user creates a series of planes to define a "slow-zone area". While in this area, the vehicle is constrained to a lower maximum velocity. The user executes short duration trajectories to highlight the effect of different velocity constraints.

Scenario 5: Quadrotor Tracking Performance

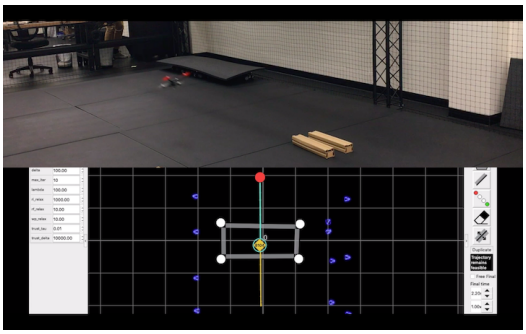
Scenario 5 shows the tracking error in SCvx-generated trajectories. This example shows the difference between guid-



(a) *Scenario 1*: executing a trajectory with boundary conditions and constraints.



(b) *Scenario 2*: mid-flight replanning of a trajectory in real-time.



(c) *Scenario 4*: quadcopter enters a slow-zone, implemented as a state-triggered constraint.

Figure 7: A snapshot of *Scenario 1* presented in Section 5, executing a trajectory with boundary conditions and constraints. Videos of these scenarios are on the website provided.

ance and navigation telemetry for position, velocity, and acceleration. Even when executing multiple trajectories in rapid succession, the position error rarely exceeds 20cm. The maximum position error is approximately 21.3cm, the minimum error is 9.7cm, and the average position error is 17.6cm. The maximum velocity error is 45.0cm/s and the maximum acceleration error is 1.395m/s^2. This is displayed in Fig. 8 and 9.

6. Conclusion & Future Work

There have been significant contributions in the area of successive convexification over the last decade, and this form of optimization-based automated trajectory planning will only continue to become more common. However, methods to abstract away the analytical problem formulation to allow

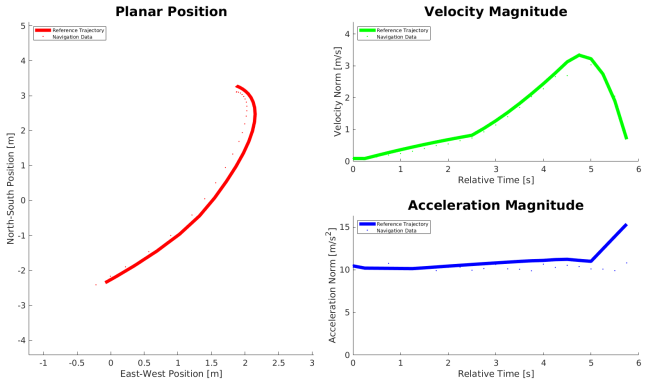


Figure 8: *Scenario 5*: Aerial drone tracking performance.

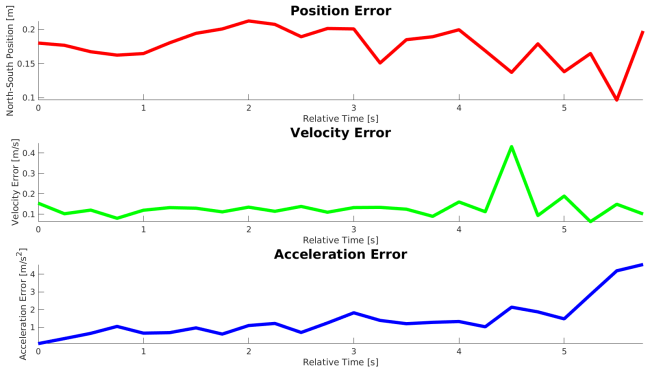


Figure 9: *Scenario 5*: Aerial drone tracking error.

a real-time feedback loop for high-level decision-making have been largely absent from the literature. This visual modeling system is a novel parsing interface that allows non-expert users to interact with the physics-based problem at a human level. It handles translating obstacles into mathematical constraints and communicating the resulting output in a simple manner. This removes the analytical step for formulating such optimization problems, incorporating users into a continuously running automation feedback loop by allowing users to add inputs to the problem and execute commands based on the resulting output displayed on the interface. This sort of continuous user integration is especially essential for SCvx problems, since they can be solved quickly and provide reliable guarantees.

Future work for the optimization interface includes exposing the mathematical representation of the problem for the user. Specifically, displaying the dual variables of the problem would allow the user to know how strongly the vehicle is pushing against a constraint and therefore influence the user's decision to change the constraint or allow more leeway. Other problem formulation properties could also be exposed to provide more control to the user. The "free final time" option could be expanded upon to allow the user to choose any of the problem parameters they wish to minimize. The user could easily switch between minimizing fuel, time, or constraint relaxation.

ACKNOWLEDGMENTS

Support for studying the convergence properties of the successive convexification framework was provided by the Office of Naval Research grants N00014-16-1-2877 and N00014-16-1-3144.

REFERENCES

[1] M. Szmuk, B. Açıkmeşe, and A. W. Berning, "Successive convexification for fuel-optimal powered landing with aerodynamic drag and non-convex constraints," in *AIAA Guidance, Navigation, and Control Conference*, Autonomous Controls Laboratory, Dept. of Aeronautics & Astronautics, University of Washington, Seattle, WA 98195, USA, January 2016.

[2] D. Dueri, B. Açıkmeşe, D. Scharf, , and M. Harris, "Customized real-time interior-point methods for onboard powered-descent guidance," *AIAA Journal of Guidance, Control and Dynamics*, vol. 30, no. 5, pp. 1353–1366, 2007.

[3] J. Peng, C. Roos, and T. Terlaky, "Self-regularity: A new paradigm for primal-dual interior-point algorithms," 2001.

[4] J. Harpold and C. A. Graves, "Shuttle entry guidance," in *Mission Planning and Analysis Division*, NASA, Houston, Texas, February 1979.

[5] X. Liu, P. Lu, and B. Pan, "Survey of Convex Optimization for Aerospace Applications," *Astrodynamics*, vol. 1, no. 1, pp. 1–23, 2017.

[6] U. Eren, A. Prach, B. B. Koçer, S. V. Raković, E. Kayacan, and B. Açıkmeşe, "Model predictive control in aerospace systems: Current state and opportunities," *Journal of Guidance, Control, and Dynamics*, vol. 40, no. 7, pp. 1541–1566, 2017.

[7] L. Blackmore, "Autonomous precision landing of space rockets," in *in Frontiers of Engineering: Reports on Leading-Edge Engineering from the 2016 Symposium*, vol. 46, 2016, pp. 15–20.

[8] P. T. Boggs and J. W. Tolle, "Sequential quadratic programming," *Acta numerica*, vol. 4, pp. 1–51, 1995.

[9] B. Fares, D. Noll, and P. Apkarian, "Robust control via sequential semidefinite programming," *SIAM Journal on Control and Optimization*, vol. 40, no. 6, pp. 1791–1820, 2002.

[10] J. Schulman, Y. Duan, J. Ho, A. Lee, I. Awwal, H. Bradlow, J. Pan, S. Patil, K. Goldberg, and P. Abbeel, "Motion planning with sequential convex optimization and convex collision checking," *The International Journal of Robotics Research*, vol. 33, no. 9, pp. 1251–1270, 2014.

[11] R. Bonalli, B. Hérissé, and E. Trélat, "Optimal control of endoatmospheric launch vehicle systems: Geometric and computational issues," *IEEE Transactions on Automatic Control*, vol. 65, no. 6, pp. 2418–2433, 2019.

[12] J. Nocedal and S. J. Wright, "Sequential quadratic programming," *Numerical optimization*, pp. 529–562, 2006.

[13] P. E. Gill and E. Wong, "Sequential quadratic programming methods," in *Mixed integer nonlinear programming*. Springer, 2012, pp. 147–224.

[14] X. Liu, Z. Shen, and P. Lu, "Entry trajectory optimization by second-order cone programming," *Journal of Guidance, Control and Dynamics*, vol. 39, no. 2, 2016.

[15] Z. Wang and Y. Lu, "Improved sequential convex programming algorithms for entry trajectory optimization," *AIAA Journal of Spacecraft and Rockets*, May 2020.

[16] M. Grant and S. Boyd, "CVX: Matlab software for disciplined convex programming, version 2.1," *http://cvxr.com/cvx*, March 2014.

[17] B. Açıkmeşe and S. R. Ploen, "Convex programming approach to powered descent guidance for Mars landing," *AIAA Journal of Guidance, Control and Dynamics*, vol. 30, no. 5, pp. 1353–1366, 2007.

[18] L. Blackmore, B. Acikmese, and D. P. Scharf, "Minimum-landing-error powered-descent guidance for mars landing using convex optimization," *Journal of Guidance, Control, and Dynamics*, vol. 33, no. 4, pp. 1161–1171, jul 2010. [Online]. Available: https://doi.org/10.2514/1.47202

[19] M. Szmuk and B. A. Açıkmeşe, "Successive convexification for 6-dof mars rocket powered landing with free-final-time," in *AIAA Guidance, Navigation, and Control Conference*, Kissimmee, FL, USA, 2018, p. 0617.

[20] M. Szmuk, T. P. Reynolds, B. A. Açıkmeşe, M. Mehran, and J. M. Carson III, "Successive convexification for 6-dof powered descent guidance with compound state-triggered constraints," in *AIAA Guidance, Navigation, and Control Conference*, San Deigo, CAL, 2019, p. 0926.

[21] K. Virtanen, H. Ehtamo, T. Raivio, and R. P. Hamalainen, "Viato-visual interactive aircraft trajectory optimization," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 29, no. 3, pp. 409–421, 1999.

[22] M. A. Rendón, F. F. Martins, and L. G. Ganimi, "A visual interface tool for development of quadrotor control strategies," *Journal of Intelligent & Robotic Systems*, pp. 1–18, 2020.

[23] J. W. S. Chong, S. Ong, A. Y. Nee, and K. Youcef-Youmi, "Robot programming using augmented reality: An interactive method for planning collision-free paths," *Robotics and Computer-Integrated Manufacturing*, vol. 25, no. 3, pp. 689–701, 2009.

[24] R. A. S. Fernandez, J. L. Sanchez-Lopez, C. Sampedro, H. Bavle, M. Molina, and P. Campoy, "Natural user interfaces for human-drone multi-modal interaction," in *2016 International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE, 2016, pp. 1013–1022.

[25] A. Sanna, F. Lamberti, G. Paravati, and F. Manuri, "A kinect-based natural interface for quadrotor control," *Entertainment Computing*, vol. 4, no. 3, pp. 179–186, 2013.

[26] A. Mashood, H. Noura, I. Jawhar, and N. Mohamed, "A gesture based kinect for quadrotor control," in *2015 International Conference on Information and Communication Technology Research (ICTRC)*. IEEE, 2015, pp. 298–301.

[27] G. Best and P. Moghadam, "An evaluation of multimodal user interface elements for tablet-based robot teleoperation," *Proc. of ARAA ACRA*, 2014.

[28] T. Pettersen, J. Pretlove, C. Skourup, T. Engedal, and T. Lokstad, "Augmented reality for programming industrial robots," in *The Second IEEE and ACM Inter-*

national Symposium on Mixed and Augmented Reality, 2003. Proceedings. IEEE, 2003, pp. 319–320.

[29] M. Ostanin and A. Klimchik, "Interactive robot programing using mixed reality," *IFAC-PapersOnLine*, vol. 51, no. 22, pp. 50–55, 2018.

[30] M. El-Jiz and L. Rodrigues, "Trajectory planning and control of a quadrotor choreography for real-time artist-in-the-loop performances," *Unmanned Systems*, vol. 6, no. 01, pp. 1–13, 2018.

[31] N. Joubert, M. Roberts, A. Truong, F. Berthouzoz, and P. Hanrahan, "An interactive tool for designing quadrotor camera shots," *ACM Transactions on Graphics (TOG)*, vol. 34, no. 6, pp. 1–11, 2015.

[32] C. Gebhardt and O. Hilliges, "Wyfiwyg: Investigating effective user support in aerial videography," *arXiv preprint arXiv:1801.05972*, 2018.

[33] T. Chakraborti, S. Sreedharan, A. Kulkarni, and S. Kambhampati, "Projection-aware task planning and execution for human-in-the-loop operation of robots in a mixed-reality workspace," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 4476–4482.

[34] P. Milgram, S. Zhai, D. Drascic, and J. Grodski, "Applications of augmented reality for human-robot communication," in *Proceedings of 1993 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'93)*, vol. 3. IEEE, 1993, pp. 1467–1472.

[35] E. A. Cappo, A. Desai, N. Dehghani, A. Momeni, and N. Michael, "Interactive online choreography for a multi-quadrotor system."

[36] P. Marion, M. Fallon, R. Deits, A. Valenzuela, C. Pérez D'Arpino, G. Izatt, L. Manuelli, M. Antone, H. Dai, T. Koolen, *et al.*, "Director: A user interface designed for robot operation with shared autonomy," *Journal of Field Robotics*, vol. 34, no. 2, pp. 262–280, 2017.

[37] C. Gebhardt, B. Hepp, T. Nägeli, S. Stevšić, and O. Hilliges, "Airways: Optimization-based planning of quadrotor trajectories according to high-level user goals," in *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, 2016, pp. 2508–2519.

[38] M. Szmuk, C. A. Pascucci, D. Dueri, and B. Açıkmeşe, "Convexification and real-time on-board optimization for agile quad-rotor maneuvering and obstacle avoidance," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Vancouver, Canada, September 2017.

[39] A. Domahidi, E. Chu, and S. Boyd, "Ecos: An socp solver for embedded systems," in *European Control Conference (ECC)*, Zurich, Switzerland, July 2013, pp. 3071–3076.

[40] M. Szmuk, C. A. Pascucci, and B. Açıkmeşe, "Real-time quad-rotor path planning for mobile obstacle avoidance using convex optimization," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Madrid, Spain, October 2018.

[41] M. Szmuk, D. Malyuta, T. P. Reynolds, M. S. Mceowen, and B. Açıkmeşe, "Real-time quad-rotor path planning using convex optimization and compound state-triggered constraints," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Autonomous Controls Laboratory, Dept. of Aeronautics & Astronautics, University of Washington, Seattle, WA 98195, USA, October 2019.

[42] Y. Mao, M. Szmuk, and B. Açıkmeşe, "Successive convexification of non-convex optimal control problems and its convergence properties," in *2016 IEEE 55th Conference on Decision and Control (CDC)*, Dec 2016, pp. 3636–3641.

[43] Y. Mao, M. Szmuk, and B. A. Açıkmeşe, "Successive convexification: A superlinearly convergent algorithm for non-convex optimal control problems," *ArXiv e-prints*, Apr 2018, arXiv:1804.06539.