2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
September 24–28, 2017, Vancouver, BC, Canada

Convexification and Real-Time On-Board Optimization for Agile Quad-Rotor Maneuvering and Obstacle Avoidance

Michael Szmuk, Carlo Alberto Pascucci, Daniel Dueri, and Behçet Açikmeşe

Abstract— **In this paper, we apply lossless and successive convexification techniques and employ real-time on-board convex optimization to perform constrained motion planning for quad-rotors. In general, this problem is challenging for real-time on-board applications due to its non-convex nature. The ability to generate feasible trajectories quickly and reliably is central to operating high-performance aerial robots in populated spaces. Motivated by our earlier research on convexification of non-convex optimal control problems, we use these convexification techniques to cast the problems into one (or a sequence of) Second-Order Cone Programming (SOCP) problem(s). In doing so, we are able to attain our solutions by leveraging modern advances in Interior Point Method (IPM) algorithms. Here, we focus on 3-degree-of-freedom trajectory generation, whereby closed-loop control is used to track a translational trajectory computed on-board at the onset of the maneuver. To the best of our knowledge, this is the first demonstration of convexification techniques used in real-time on-board trajectory generation for high-performance quad-rotor flight. We present two example scenarios: (1) a case where lossless convexification is used to increase the available control envelope, thus enabling an agile flip maneuver, and (2) a case where both convexification techniques are used to compute a path through a flight space containing ellipsoidal keep-out zones. We present flight demonstration results obtained using the Autonomous Control Laboratory's (ACL's) custom quad-rotor platforms and SOCP optimization software. Additionally, computation timing statistics for the example scenarios obtained using a series of mobile ARM and Intel processors show a minimum mean computation time of 36.5 and 122.2 milliseconds, respectively.**

I. Introduction

In this paper, our main contribution is to present preliminary experimental results on the application of lossless and successive convexification techniques to real-time on-board trajectory generation for quad-rotor systems. To demonstrate these techniques, we present two scenarios. In the first, we utilize *lossless* convexification to perform an agile flip maneuver. In doing so, we take advantage of a non-convex region of the control space, thus enabling the vehicle to exploit a portion of its feasible flight envelope that could not be used with existing convex optimization techniques. In the second, we employ both *lossless* and *successive* convexification to compute a feasible trajectory through a non-convex flight space populated with cylindrical no-fly-zones. These problems present some of the key challenges inherent in autonomous aerial robotic applications.

As quad-rotor and aerial robotics systems have come into prominence, their missions have increased in complexity. In many such applications, aerial platforms would be required to operate in cluttered environments, often in close proximity to humans. Thus, the ability to move in a constrained flight space while complying with environmental constraints is key to ensuring that aerial robotics systems can interact with humans in a safe and efficient manner. Specifically, these missions depend on the ability of the vehicles to transition from one state to another in a dynamically feasible way. Ideally, this can be done without sacrificing the performance that is inherent to the quad-rotor system. This topic has been addressed in a variety of ways, with implementations ranging from off-board precomputed trajectories to on-board real-time solutions.

Historically, several path planning approaches have been proposed in the literature. Classical methods based on potential fields [1] make use of repulsive and attractive forces based on objects' assigned polarity. Although intuitive and easy to implement, these methods tend to generate conservative trajectories, and often produce undesirable equilibrium points away from the prescribed destination [2]. In [3], Mixed Integer Quadratic Programming (MIQP) is used to generate feasible trajectories and avoid obstacles for a small spacecraft in the context of rendezvous, docking, and proximity operation with the International Space Station. However, MIQPs are generally NP-hard and quickly become computationally intractable for on-board and real-time implementations. In [4], a hierarchical approach combines Model Predictive Control (MPC) and Hybrid MPC for quad-rotor stabilization and path planning with obstacle avoidance, and in [5], MPC is used to generate optimal trajectories to steer a quad-rotor from its initial state to a desired one, but the computations were done off-board on a standard desktop computer. Moreover, agile maneuvering often requires controls from the extremal points of the (possibly non-convex) feasible control space. Further, high-performance maneuvers typically traverse enough of the state-space that linearization about a single condition results in poor performance, or even instability. To overcome these issues, nonlinear MPC solvers [6] and methods like [7], [8] have been shown to be effective on multi-rotor vehicles equipped with high performance processors. Other approaches address the problem by exploiting the differential flatness property of the system to generate minimum snap trajectories [9]. This technique can also be embedded on-board and in real-time in a receding horizon scheme [10] or in a trajectory segmentation framework in which a different controller is designed for each flight phase [11].

Our approach focuses on leveraging the speed and reliabil-

The authors are with the Autonomous Controls Laboratory of the Department of Aeronautics and Astronautics, University of Washington, Seattle, WA 98105, USA {mszmuk, carloal, dandueri, behcet}@uw.edu

ity of convex optimization techniques to generate quad-rotor trajectories in a computationally tractable manner. This is increasingly true due to recent advancements in powerful customized convex optimization solvers [12]–[15]. While other successful applications of convex optimization have used lower-order methods to similar ends [13], [16], our use of second-order methods allows us to handle a broader set of constraints. Although quad-rotor trajectory optimization problems are inherently non-convex, their dynamics and control constraints bear a strong resemblance to those encountered in the planetary powered rocket landing problem. Consequently, the lossless convexification technique developed for the latter application [17], [18], and that was later generalized to a class of linear systems [19], [20], can be applied to the former. Lossless convexification of non-convex *control* constraints is important because it captures two key characteristics of quad-rotor motion in a convex optimization framework. First, it enables the application of a lower bound on the norm of the commanded thrust vector. This primarily ensures that the rotors continue to spin, and that the quad-rotor maintains attitude control authority throughout the trajectory. Second, it enables the use of the full tilt envelope, thus allowing the vehicle's thrust vector to point beyond 90° off-vertical.

Trajectory optimization for motion planning problems with non-convex *state* constraints require additional convexification in order to be solved in a convex optimization framework. For such scenarios, successive convexification can be employed to solve the problem as a sequence of convex optimization problems. This convexification strategy has been used for non-convex rocket landing problems [21], [22] and hypersonic re-entry scenarios [23], and recent theoretical results on its convergence properties guarantee that if the process converges to a feasible solution, then the solution is a local optimum of the original non-convex problem [24]–[26]. In the case of quad-rotors, successive convexification is important because it enables us to solve a non-convex path planning problem using convex optimization methods well suited for on-board real-time use.

Using these convexification techniques, the problems outlined at the beginning of this section are cast into a second-order cone programming (SOCP) convex optimization framework. The resulting SOCPs are solved using fast and reliable Interior Point Method (IPM) algorithms. We emphasize that all computations presented in this paper were performed in real-time and on-board the vehicle, whereby the trajectories were computed once at the onset of each maneuver, and thereafter tracked using closed-loop control.

The remainder of this paper is organized as follows. In Section II, we give a brief primer on lossless and successive convexification. In Section III, we describe the hardware used in our experiments, and we present a high-level system architecture to illustrate how the methods described herein interact with the quad-rotor system. In Section IV, we outline the problem formulations of the two aforementioned scenarios. Experimental results are presented in Section V, and concluding remarks are given in Section VI.

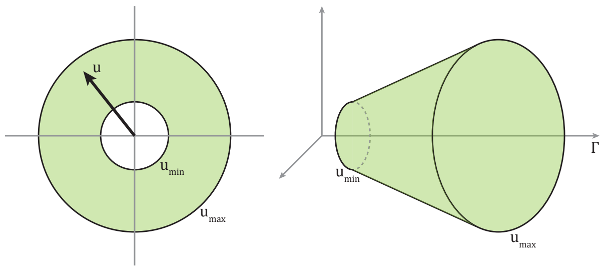

Fig. 1. A geometric interpretation of lossless convexification for non-convex control constraints. (Left): Non-convex set of feasible controls; (Right): Convexified set of feasible controls

II. Convexification

A. Lossless Convexification

Lossless convexification is a technique used to handle non-convex *control* constraints in an otherwise convex optimal control problem. To illustrate the technique, consider a fixed-time optimal control problem with state variable $\mathbf{x}(t) \in \mathbb{R}^n$ and control variable $\mathbf{u}(t) \in \mathbb{R}^3$. Suppose that our objective is to minimize the integral of $\|\mathbf{u}(t)\|_2$ subject to the following constraints:

$$0 < u_{min} \leq \|\mathbf{u}(t)\|_2 \leq u_{max} \tag{1a}$$

$$\|\mathbf{u}(t)\|_2 \cos(\theta_{max}) \leq \hat{\mathbf{n}}^T \mathbf{u}(t) \tag{1b}$$

where u_{min} and u_{max} are, respectively, the minimum and maximum allowable magnitudes of $\mathbf{u}(t)$, and $\theta_{max} \in (0°, 180°]$ is the maximum angle $\mathbf{u}(t)$ is allowed to deviate from $\hat{\mathbf{n}}$. Furthermore, suppose that $\mathbf{x}(t) \in \mathbb{X}$, and that $\mathbb{X}$ is a convex set. Clearly, Eqs. 1 define a non-convex control domain, thus rendering the optimal control problem non-convex. Now, consider the relaxed problem whereby we introduce a slack variable, $\Gamma(t)$, and seek to minimize the integral of $\Gamma(t)$ subject to $\mathbf{x}(t) \in \mathbb{X}$ and the following constraints:

$$0 < u_{min} \leq \Gamma(t) \leq u_{max} \tag{2a}$$

$$\|\mathbf{u}(t)\|_2 \leq \Gamma(t) \tag{2b}$$

$$\Gamma(t) \cos(\theta_{max}) \leq \hat{\mathbf{n}}^T \mathbf{u}(t) \tag{2c}$$

Note that due to Eqs. 2, the relaxed optimal control problem is now convex, although it is not obvious that the inequality in Eq. 2b remains tight. A geometric interpretation of this relaxation is shown in Figure 1.

Simply stated, lossless convexification guarantees that if a feasible solution exists, the optimal solution to the (convex) relaxed problem recovers the optimal solution of the original non-convex problem. That is, we are guaranteed that the inequality in Eq. 2b remains tight, and consequently, we are able to apply powerful convex optimization algorithms to solve the original non-convex problem.

The reader is referred to references [17], [20] for more details on lossless convexification of the thrust lower bound, and to [27] for lossless convexification of the pointing constraint when $\theta_{\max} \in (90°, 180°)$.

B. Successive Convexification

Successive convexification is a technique by which a non-convex optimal control problem is solved by solving a sequence of convex sub-problems. The original problem may have non-convex dynamics, and may contain non-convexities in both state equality and inequality constraints. Each sub-problem is derived from the original problem by linearizing the former about the solution from the previous iteration. The process is relatively easy to initialize (e.g. see [21], [22]). Additionally, the problem is augmented with trust regions and virtual controls in order to avoid artificial unboundedness and infeasibility, respectively.

Recent theoretical results guarantee that if the solution process converges to a feasible solution, then the solution is a feasible local optimum of the original non-convex problem. However, in general, if no feasible solution is found, we are not guaranteed that the original problem is infeasible, and conversely, if the original problem is feasible, we are not guaranteed to find a feasible solution. Nevertheless, under mild assumptions, successive convexification performs well in practice.

The reader is referred to references [24]–[26] for more details on successive convexification.

III. System Description

A. Hardware

The experimental results described in this paper were generated using the Autonomous Controls Laboratory's (ACL's) custom-built quad-rotor platform, shown in Figure 2. The platform consists of a rugged custom-designed carbon fiber frame, a 2000 mAh LiPo battery, and four brushless DC motors. The on-board electronics were custom designed to house a Gumstix Overo Computer-On-Module that runs an ARM Cortex-A7 clocked at 750 MHz, has 512 MB of RAM, and uses an IEEE 802.11g compliant WiFi chip for network connectivity. All of the on-board and ground control software was written in C++, and was custom made for ACL's quad-rotor platform.

For indoor positioning, the ACL is equipped with an OptiTrack motion capture (MoCap) system that provides position data at millimeter-level accuracy using the Motive tracking software application. The system is comprised of Prime 17W and Prime 41 cameras, and can provide position and attitude data at up to 180 Hz.

B. SOCP Solution Algorithm and Software

The objective of our SOCP software is to quickly and reliably solve optimization problems related to our path planning approach. To this end, we utilize a custom C++ IPM solver that is capable of efficiently solving SOCP optimization problems [12], [28]. IPMs have polynomial time complexity and are guaranteed to find an optimal solution if a feasible solution exists. Conversely, if a problem is infeasible, IPMs provide a certificate of infeasibility in polynomial time complexity. Further, we make use of a highly efficient in-house C++ parser that easily transforms human-readable optimization problems (such as Problem 1) into a format that the IPM is expecting. Together, these tools enable us to solve convex optimization problems in real-time onboard the quad-rotors.

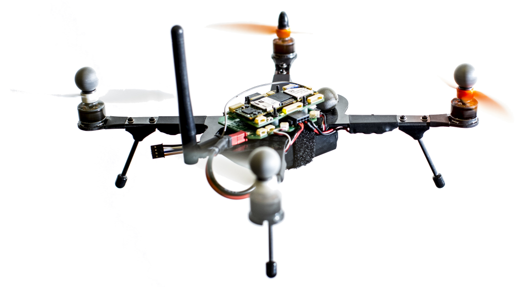

Fig. 2. ACL's custom quad-rotor platform. The vehicle is approximately 240 mm long from motor to motor and can hover for over 20 minutes.

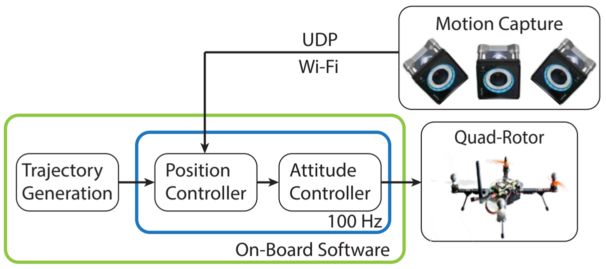

Fig. 3. This plot gives a high-level illustration of the system architecture. The content inside the green rectangle resides on-board the vehicle, and the content in the blue rectangle executes continuously at a high frequency.

C. System Architecture

Figure 3 provides a high-level illustration of our system architecture. As shown, the aforementioned SOCP software was used to generate trajectories on-board the vehicle. The optimization routine was executed once, after which its output was used as a reference for a feedback controller that we refer to as the *position* controller. This controller is charged with tracking the commanded positions, velocities, and accelerations of the generated trajectory. The position controller generates commands for the *attitude* controller, which actuates the motors in order to track the commanded attitude. The control loop was executed at 100 Hz, and position and velocity feedback were uplinked to the vehicle over the lab's WiFi network using UDP.

IV. Problem Formulation

A. Agile Flip Maneuver

To demonstrate the benefits afforded by lossless convexification, we present a fixed-final-time optimal control problem that performs an agile flip maneuver. In what follows, we use t to denote time, and the subscripts i, c, and f to denote the initial, intermediate, and final temporal points, respectively. We model the vehicle using 3-degree-of-freedom (3-DoF) translational dynamics, and denote the vehicle's position, velocity, and commanded acceleration as $\mathbf{r} \in \mathbb{R}^3$, $\mathbf{v} \in \mathbb{R}^3$, and $\mathbf{u} \in \mathbb{R}^3$, respectively.

Problem 1:

$$\underset{\mathbf{u}(t),\Gamma(t)}{\text{minimize}} \quad \int_0^{t_f} \Gamma^2(t)dt$$

subject to:

$$\mathbf{r}(0) = \mathbf{r}_i \qquad \mathbf{v}(0) = \mathbf{v}_0 \qquad \mathbf{u}(0) = g\mathbf{e}_3$$
$$\mathbf{r}(t_c) = \mathbf{r}_c \quad \mathbf{e}_1^T\mathbf{v}(t_c) = 0$$
$$\mathbf{e}_3^T\mathbf{v}(t_c) = 0$$
$$\mathbf{r}(t_f) = \mathbf{r}_f \qquad \mathbf{v}(t_f) = \mathbf{v}_f \quad \mathbf{u}(t_f) = g\mathbf{e}_3$$

$$\dot{\mathbf{r}}(t) = \mathbf{v}(t)$$
$$\dot{\mathbf{v}}(t) = \mathbf{u}(t) - g\mathbf{e}_3$$

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

This formulation is used to keep to optimization problem tractable, while closed-loop control is used to address the higher-order dynamics present in the physical system. Consequently, some conservatism must be used in selecting the 3-DoF problem parameters to ensure the closed-loop control retains sufficient control authority.

We use $\mathbf{e}_j$ to denote the unit vector along the j^{th}-axis, and g to denote gravitational acceleration. Unless otherwise stated, all remaining parameters are constants that will be numerically specified in Section V-A. The optimal control problem is stated in Problem 1. Note that unlike the minimum fuel objective used in Section II-A, here we use a minimum-energy objective function. This objective still allows us to use lossless convexification, and ensures that the trajectory is smoother, and thus easier to follow. To implement this problem using a numerical solver, the problem is discretized assuming a first-order-hold on the control variable $\mathbf{u}(t)$. As with our choice of the objective function, we assume a first-order-hold instead of a zero-order-hold in order to make the trajectory (and the control input) smoother and easier to track.

B. Obstacle Avoidance

To demonstrate the capabilities of successive convexification, we present a fixed-final-time optimal control problem where the objective is to fly from one point to another while avoiding known obstacles (i.e., obstacle locations, shapes, and sizes are known a-priori). First, we define our flight space using a set of half-spaces (i.e. floor, ceiling, etc.). Next, we populate our flight space with three-dimensional ellipsoidal keep-out zones, described by the following non-convex constraint

$$\|H_j(\mathbf{r}(t) - \mathbf{p}_j)\|_2 \geq R_j \quad \forall j \in \mathbb{J} \tag{3}$$

where $\mathbf{p}_j$ defines the center of the j^{th} ellipsoid, and where H_j and R_j define the shape and size of the the j^{th} ellipsoid. Note that H_j is assumed to be symmetric positive semi-definite, and that the number of keep-out zones is given by $\mathbf{card}(\mathbb{J})$. We assume that none of the keep-out zones intersect, and that each keep-out zone is either entirely contained within the flight space, or its center, $\mathbf{p}_j$, is outside the flight zone. Next, we introduce a relaxation term, $\nu_j \geq 0$, to the linearized version of Eq. 3 (see Eq. 4). This relaxation allows for violations of the keep-out zones during iterations where the solution has not yet converged, thus allowing the sub-problem to remain feasible and the convergence process to continue. The dynamics and control constraints of the quad-rotor are assumed to be the same as in Section IV-A (see Problem (2)), and thus, we retain the slack variable Γ introduced by the lossless convexification.

Before we present the algorithm, we note that a double integrator subject to acceleration constraints can follow any twice-continuously-differentiable path provided that it is given a *long enough* time to do so. Thus, it follows that if the problem stated here admits a feasible path, then there exists a time t_f^* such that for $t_f \geq t_f^*$ a feasible solution can be found.

The remainder of our solution strategy consists of solving Problem 2 in accordance to the steps outlined in Algorithm 1. Values for parameters that have not been specified will be provided in Section V-B.

V. Experimental Results

In this section we present experimental flight results to (1) show that the proposed problem formulations capture enough of the dynamics to be practically useful, and (2) that the solutions to these optimal control problems can be computed quickly and reliably on embedded processors on-board the quad-rotors.

Algorithm 1:

```
Step 1: Select a final time, $t_f > 0$.
Step 2: Solve Problem 2 using $t_f$ without Eq. 4.
  Step 2.1: If feasible, apply successive convexification
      (as in [24]) to solve Problem 2 using $t_f$.
    Step 2.1.1: If $\sup_{t\in 0,t_f} \|\mathbf{r}^{k+1}(t) - \mathbf{r}^k(t)\|_\infty \leq r_{tol}$ for
        iterations $k$ and $(k+1)$, then Exit.
    Step 2.1.2: Else if number of iterations exceeds
        $i_{max}$ reached, return to Step 1, and set $t_f$ to
        $\alpha t_f$, $\alpha > 1$.
  Step 2.2: Else, return to Step 1, and set $t_f$ to
      $\alpha t_f$, $\alpha > 1$.
```

Problem 2:

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

subject to:

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

$$\dot{\mathbf{r}}^k(t) = \mathbf{v}^k(t)$$
$$\dot{\mathbf{v}}^k(t) = \mathbf{u}^k(t) - g\mathbf{e}_3$$

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

$$x_{min} \le \mathbf{e}_1^T\mathbf{x}^k(t) \le x_{max}$$
$$y_{min} \le \mathbf{e}_2^T\mathbf{x}^k(t) \le y_{max}$$
$$z_{min} \le \mathbf{e}_3^T\mathbf{x}^k(t) \le z_{max}$$

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

$$\nu_j \ge 0$$
$$H_j \succeq 0$$
$$\Delta\mathbf{r}^{k,j}(t) \triangleq (\mathbf{r}^{k-1}(t) - \mathbf{p}_j)$$
$$\delta\mathbf{r}^k(t) \triangleq \mathbf{r}^k(t) - \mathbf{r}^{k-1}(t)$$
$$\xi^{k,j}(t) \triangleq \|H_j\Delta\mathbf{r}^{k,j}(t)\|_2$$
$$\boldsymbol{\zeta}^{k,j}(t) \triangleq \frac{H_j^T H_j \Delta\mathbf{r}^{k,j}(t)}{\|H_j\Delta\mathbf{r}^{k,j}(t)\|_2}$$

$$\xi^{k,j} + \left[\boldsymbol{\zeta}^{k,j}(t)\right]^T \delta\mathbf{r}^k(t) \ge R_j - \nu_j \qquad (4)$$

A. Agile Flip Maneuver

The parameters used to implement Problem 1 are presented in Table I. They were selected such that the vehicle starts at the origin, passes through a waypoint that is up and to the right while moving with an unspecified velocity in the Y-direction, and then returns to a point 3 meters away from the origin along the Y-axis. The final time was chosen such that the maximum vehicle tilt angle exceeds 90°. 23 time discretization points were used in order to retain sufficient time sampling while keeping the runtime sufficiently low.

The commanded and actual trajectories are shown in Figure 4. The thrust tilt and magnitude profiles are shown in Figure 5. The commanded tilt angle can be seen to reach the 100° limit between 0.6 and 0.8 seconds.

As can be seen, the tracking error grows as the trajectory proceeds. We believe this is largely due to Problem 1's assumption that translational and attitude control efforts can be decoupled. This assumption relies heavily on the quick response time of the attitude loop and on the effectiveness of the feedback controller. Additionally, it is possible that a more conservative choice for u_{min} and u_{max} would result in better closed-loop performance, as the controller would have more thrust to correct for deviations. Further work is required to improve closed-loop three-dimensional trajectory tracking.

TABLE I
PARAMETERS FOR AGILE FLIP MANEUVER

Parameter	Value	Units	Parameter	Value	Units
t_c	0.7	s	$\mathbf{r}_i$	$\begin{bmatrix}0 & 0 & 0\end{bmatrix}^T$	m
t_f	1.4	s	$\mathbf{r}_c$	$\begin{bmatrix}1 & 1.5 & 1\end{bmatrix}^T$	m
g	9.81	m/s^2	$\mathbf{r}_f$	$\begin{bmatrix}0 & 3 & 0\end{bmatrix}^T$	m
u_{min}	0.6	m/s^2	$\mathbf{v}_i$	$\begin{bmatrix}0 & 0 & 0\end{bmatrix}^T$	m/s
u_{max}	23.2	m/s^2	$\mathbf{v}_f$	$\begin{bmatrix}0 & 0 & 0\end{bmatrix}^T$	m/s
θ_{max}	100	°			

TABLE II
TIMING STATISTICS FOR AGILE FLIP MANEUVER (IN [MS])

Processor	Min	Max	Med.	Mean	Std. Dev.
Overo	610.32	618.35	612.17	612.22	1.25
BBB	330.06	410.83	331.80	335.05	11.02
Edison	218.97	229.57	219.21	219.39	1.05
Joule	36.14	41.06	36.45	36.48	0.24

The timing statistics of Problem 1 are given in Table II. All times are given in milliseconds. The statistics are given for four systems: (1) the Gumstix Overo with an ARM Cortex-A7 clocked at 750 MHz (flown on-board the vehicle), (2) the BeagleBone Black (BBB) with an ARM Cortex-A8 clocked at 1 GHz, (3) the Intel Edison with a dual-core Atom clocked at 500 MHz, and (4) the Intel Joule with a quad-core Atom clocked at 1.7 GHz. All computations were performed using double precision arithmetic.

B. Obstacle Avoidance

The parameters used to implement Algorithm 1 and Problem 2 are provided in Table III. Vehicle parameters

TABLE III
PARAMETERS FOR OBSTACLE AVOIDANCE

Parameter	Value	Units
t_f	2.0	s
α	1.5	-
i_{max}	5	-
r_{tol}	0.05	m
H_j	$\mathbf{diag}(\begin{bmatrix}1 & 1 & 0\end{bmatrix})$	-
w	$1e^{-5}$	-

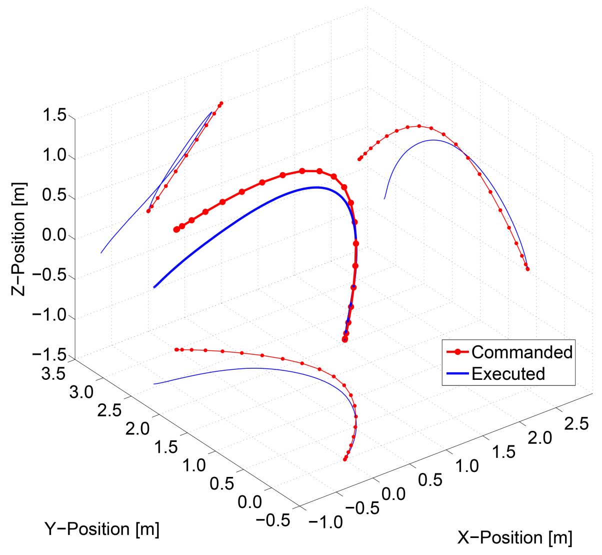

Fig. 4. The agile flip maneuver trajectory. The red line indicates the commanded trajectory, and the blue represents the flown trajectory. The solid points along the red line indicate the discretization points of the optimization problem, and the thin lines represent projections of the three-dimensional profiles. The motion is from right to left.

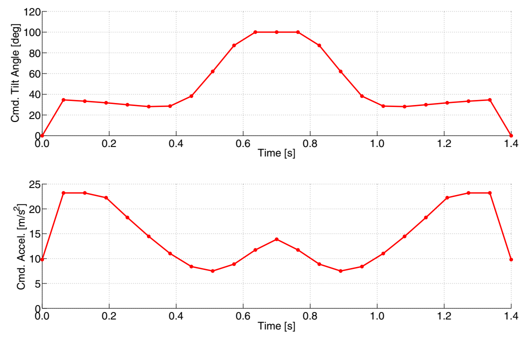

Fig. 5. The top and bottom plots show the commanded tilt angles and accelerations as a function of time, respectively. The solid points represent the discrete time instances solved by the onboard optimizer.

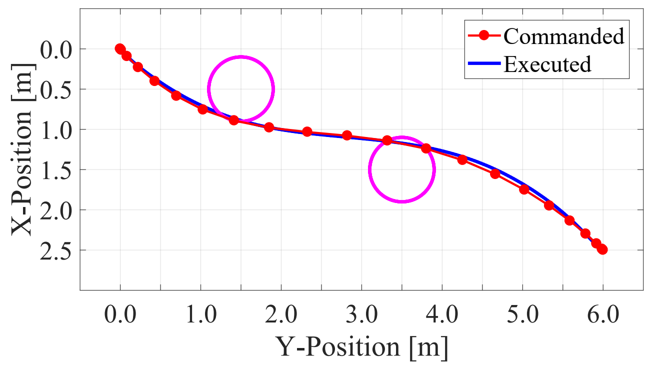
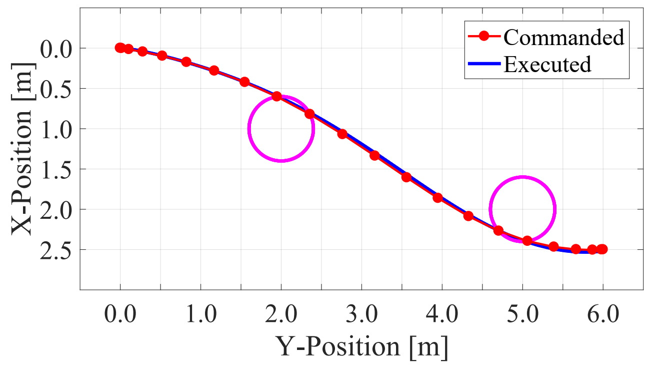

Fig. 6. The top figure shows the commanded and executed trajectories in red and blue, respectively. The keep-out zones are shown as circles, and the motion is from left to right. The bottom figure shows the same information for the second obstacle configuration.

were provided in Table I. In this scenario, we present the algorithm with two obstacle configurations consisting of the same two obstacles placed in different locations within the flight space. Flight area boundaries were not activated in this scenario and are thus omitted for brevity. The vehicle is required to traverse 6 meters by 2.5 meter space in 2 seconds while maintaining a constant altitude, starting and ending in a hover condition. This problem was discretized with 22 time discretization points for the same reasons mentioned in Section V-A.

The computed and executed trajectories are shown in relation to the obstacles in Figure 6. As seen in the figure, the algorithm guides the vehicle to the right of the first obstacle and to the left of the second obstacle for the first configuration, and does the opposite for the second configuration. Despite being commanded to tilt angles in excess of 46 degrees, the closed-loop controller maintained the tracking error below 5 centimeters throughout the entirety of both trajectories.

Note that the discretization of the problem results in clipping of the keep-out-zones. This can be mitigated through a combination of enlarging the radius of the keep-out zones, increasing the number of discretization points, and imposing a sufficiently low maximum velocity constraint on the vehicle. Lastly, timing statics for Algorithm 1 are given in Table IV for the same systems referenced in Table II. It is notable that the Intel processors largely outperformed the ARM processors without a significant increase in power consumption. Moreover, the compact size of the Intel processors makes them ideal for aerial robotic applications, and

TABLE IV
TIMING STATISTICS FOR OBSTACLE AVOIDANCE MANEUVER (IN [MS])

Processor	Min	Max	Med.	Mean	Std. Dev.
Overo	2065	2089	2069	2069	2.93
BBB	1209	1259	1214	1224	18.31
Edison	753.7	763.7	754.1	754.7	2.09
Joule	120.4	131.2	122.5	122.2	1.12

the performance of the Joule makes real-time successive convexification attainable.

VI. Conclusion

In this paper, we have demonstrated that numerical optimization techniques based on convex optimization are well-suited for applications that require real-time on-board motion planning in the presence of non-convex constraints. First, an agile flip maneuver that required the use of non-convex control constraints was planned on-board the vehicle in real-time via lossless convexification. Then, the quad-rotor computed trajectories that avoided cylindrical obstacles by utilizing successive convexification techniques. Trajectory generation runtimes in Table IV suggest that obstacle avoidance trajectories can be obtained at approximately 10 Hz, and thus could be linked with a computer vision toolkit that detects (potentially moving) obstacles in real-time. Future work will focus on further improving the computational efficiency of successive convexification algorithms and acquiring theoretical bounds on its convergence rate.

References

[1] S. Haddadin, R. Belder, and A. Albu-Schäffer, "Dynamic motion planning for robots in partially unknown environments," *IFAC Proceedings Volumes*, vol. 44, no. 1, pp. 6842–6850, 2011.

[2] J.-H. Chuang, "Potential-based modeling of three-dimensional workspace for obstacle avoidance," *IEEE Transactions on Robotics and Automation*, vol. 14, no. 5, pp. 778–785, Oct 1998.

[3] A. Richards, T. Schouwenaars, J. P. How, and E. Feron, "Spacecraft trajectory planning with avoidance constraints using mixed-integer linear programming," *Journal of Guidance, Control, and Dynamics*, vol. 25, no. 4, pp. 755–764, 2002.

[4] A. Bemporad, C. Pascucci, and C. Rocchi, "Hierarchical and hybrid model predictive control of quadcopter air vehicles," {*IFAC*} *Proceedings Volumes*, vol. 42, no. 17, pp. 14 – 19, 2009, 3rd {IFAC} Conference on Analysis and Design of Hybrid Systems. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1474667015307308

[5] M. W. Mueller and R. D'Andrea, "A model predictive controller for quadrocopter state interception," in *2013 European Control Conference (ECC)*, July 2013, pp. 1383–1389.

[6] B. Houska, H. J. Ferreau, and M. Diehl, "An auto-generated real-time iteration algorithm for nonlinear mpc in the microsecond range," *Automatica*, vol. 47, no. 10, pp. 2279–2285, 2011.

[7] M. Kamel, K. Alexis, M. Achtelik, and R. Siegwart, "Fast nonlinear model predictive control for multicopter attitude tracking on so(3)," in *2015 IEEE Conference on Control Applications (CCA)*, Sept 2015, pp. 1160–1166.

[8] M. Neunert, C. de Crousaz, F. Furrer, M. Kamel, F. Farshidian, R. Siegwart, and J. Buchli, "Fast nonlinear model predictive control for unified trajectory optimization and tracking," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, May 2016, pp. 1398–1404.

[9] D. Mellinger and V. Kumar, "Minimum snap trajectory generation and control for quadrotors," in *2011 IEEE International Conference on Robotics and Automation*, May 2011, pp. 2520–2525.

[10] M. Watterson and V. Kumar, "Safe receding horizon control for aggressive mav flight with limited range sensing," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sept 2015, pp. 3235–3240.

[11] S. Liu, M. Watterson, S. Tang, and V. Kumar, "High speed navigation for quadrotors with limited onboard sensing," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, May 2016, pp. 1484–1491.

[12] D. Dueri, J. Zhang, and B. Açikmese, "Automated custom code generation for embedded, real-time second order cone programming," in *19th IFAC World Congress*, 2014, pp. 1605–1612.

[13] J. Mattingley and S. Boyd, "Cvxgen: A code generator for embedded convex optimization," *Optimization and Engineering*, vol. 13, no. 1, pp. 1–27, 2012.

[14] A. Domahidi, E. Chu, and S. Boyd, "ECOS: An SOCP solver for Embedded Systems," in *Proceedings European Control Conference*, 2013.

[15] A. Domahidi, A. U. Zgraggen, M. N. Zeilinger, M. Morari, and C. N. Jones, "Efficient interior point methods for multistage problems arising in receding horizon control," in *Decision and Control (CDC), 2012 IEEE 51st Annual Conference on*. IEEE, 2012, pp. 668–674.

[16] J. L. Jerez, P. J. Goulart, S. Richter, G. A. Constantinides, E. C. Kerrigan, and M. Morari, "Embedded online optimization for model predictive control at megahertz rates," *IEEE Transactions on Automatic Control*, vol. 59, no. 12, pp. 3238–3251, 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. Açıkmeşe, and D. P. Scharf, "Minimum-landing-error powered-descent guidance for mars landing using convex optimization," *AIAA Journal of Guidance, Control and Dynamics*, vol. 33, no. 4, pp. 1161–1171, 2010.

[19] B. Açıkmeşe and L. Blackmore, "Lossless convexification of a class of optimal control problems with non-convex control constraints," *Automatica*, vol. 47, no. 2, pp. 341–347, 2011.

[20] M. Harris and B. Açıkmeşe, "Lossless convexification of non-convex optimal control problems for state constrained linear systems," *Automatica*, vol. 50, no. 9, pp. 2304–2311, 2014.

[21] M. Szmuk, B. A. Açıkmeşe, and A. W. B. Jr., "Successive convexification for fuel-optimal powered landing with aerodynamic drag and non-convex constraints," in *AIAA Guidance, Navigation, and Control Conference*, 2016, p. 0378.

[22] M. Szmuk, U. Eren, and B. A. Açıkmeşe, "Successive convexification for mars 6-dof powered descent landing guidance," in *AIAA Guidance, Navigation, and Control Conference*, 2017, p. 1500.

[23] 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, pp. 227–241, 2015.

[24] 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.

[25] Y. Mao, M. Szmuk, and B. Acikmese, "Successive convexification of non-convex optimal control problems and its convergence properties," *ArXiv e-prints*, Aug. 2016, arXiv:1608.05133.

[26] Y. Mao, D. Dueri, M. Szmuk, and B. Açıkmeşe, "Successive convexification of non-convex optimal control problems with state constraints," *ArXiv e-prints*, Jan. 2017, arXiv:1701.00558.

[27] J. M. Carson, B. Açıkmeşe, L. Blackmore, and A. A. Wolf, "Capabilities of convex powered-descent guidance algorithms for pinpoint and precision landing," in *Aerospace Conference, 2011 IEEE*. IEEE, 2011, pp. 1–8.

[28] D. Dueri, B. Açıkmeşe, D. P. Scharf, and M. W. Harris, "Customized real-time interior-point methods for onboard powered-descent guidance," *Journal of Guidance, Control, and Dynamics*, pp. 1–16, 2016.