Mixed-Integer Quadratic Program Trajectory Generation for Heterogeneous Quadrotor Teams

Daniel Mellinger, Alex Kushleyev, and Vijay Kumar

***Abstract*— We present an algorithm for the generation of optimal trajectories for teams of heterogeneous quadrotors in three-dimensional environments with obstacles. We formulate the problem using mixed-integer quadratic programs (MIQPs) where the integer constraints are used to enforce collision avoidance. The method allows for different sizes, capabilities, and varying dynamic effects between different quadrotors. Experimental results illustrate the method applied to teams of up to four quadrotors ranging from 65 to 962 grams and 21 to 67 cm in width following trajectories in three-dimensional environments with obstacles with accelerations approaching $1g$.**

I. Introduction

Multi-rotor aerial vehicles have become increasingly popular robotic platforms because of their mechanical simplicity, dynamic capabilities, and suitability for both indoor and outdoor environments. In particular, there have been many recent advances in the design [6], control [9] and planning [7] for quadrotors, rotorcrafts with four rotors. In this paper we present a method for generating optimal trajectories for heterogeneous quadrotor teams like those shown in Fig. 1 in environments with obstacles.

Trajectories that quadrotors can follow quickly and accurately should be continuous up to the third derivative of position (or C^3). This is because, for quadrotors, discontinuities in lateral acceleration require instantaneous changes in roll and pitch angles and discontinuities in lateral jerk require instantaneous changes in angular velocity. Finding C^3 trajectories requires planning in a high-dimensional search space which is impractical for methods using reachability algorithms [5], incremental search techniques [8] or LQR-tree-based searches [17]. The problem is exacerbated when planning for multiple vehicles as this further expands the dimension of the search space.

This paper builds on our own previous work [10] in which we showed that the dynamic model for the quadrotor is differentially flat. We used this fact to derive a trajectory generation algorithm that allows us to naturally embed constraints on desired positions, velocities, accelerations, jerks and inputs while satisfying requirements on smoothness of the trajectory. We extend that method in this work to include multiple quadrotors and obstacles. The method allows for different sizes, capabilities, and varying dynamic effects between different quadrotors. We enforce collision avoidance using integer constraints which transforms our quadratic program (QP) from [10] into a mixed-integer quadratic program (MIQP).

D. Mellinger and V. Kumar are with the Department of Mechanical Engineering and Applied Mechanics, A. Kushleyev is with the Department of Electrical and Systems Engineering. University of Pennsylvania, Philadelphia, PA 19104, USA {dmel, akushley, kumar}@seas.upenn.edu

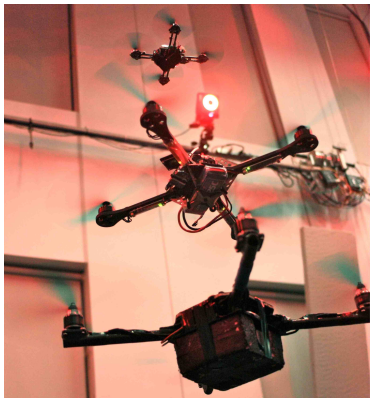


Fig. 1. The kQuad65 (top), the Asctec Hummingbird [1] (middle), and the kQuad1000 (bottom).

Our work also draws from the extensive literature on mixed-integer linear programs (MILPs) and their application to trajectory planning from Schouwenaars et al. [14], [15], [16], [13]. This body of work demonstrates the power and flexibility of integer constraints in similar trajectory planning problems for both fixed-wing aerial vehicles and rotorcraft. A key difference in our approach is that we use piece-wise smooth polynomial functions to synthesize trajectories in the flat output space. Using piece-wise smooth polynomial functions allows us to enforce continuity between waypoints up to any desired derivative of position. Another difference in our work from this previous work on trajectory generation is the use of quadratic cost function resulting in a MIQP as opposed to a MILP.

The organization of the paper is as follows. First, we present a model for the quadrotor dynamics and a controller for following specified trajectories in Secs. II and III. These two sections are mostly drawn from our previous work [11], [10]. Next, we present our trajectory generation method for a single quadrotor in Sec. IV and its extension to heterogeneous quadrotor teams in Sec. V. In Sec. VI, we present experimental results for teams of up to four quadrotors ranging from 65 to 962 grams and 21 to 67 cm in width shown in Fig. 1 following trajectories in three-dimensional environments with obstacles with accelerations approaching $1g$. Finally, in Sec.VII, we offer some concluding remarks on this approach.

II. Model

The coordinate systems including the world frame, $\mathcal{W}$, and body frame, $\mathcal{B}$, as well as the propeller numbering convention

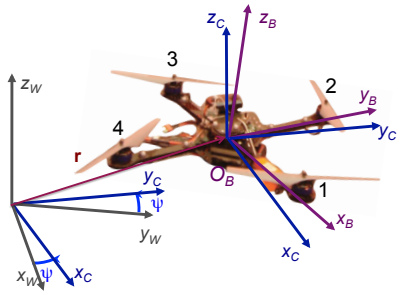


Fig. 2. The reference frames and propeller numbering convention.

for the quadrotor are shown in Fig. 2. We also use $Z-X-Y$ Euler angles to define the roll, pitch, and yaw angles (ϕ, θ, and ψ) as a local coordinate system. The rotation matrix from $\mathcal{B}$ to $\mathcal{W}$ is given by ${}^{W}R_B = {}^{W}R_C {}^{C}R_B$ where ${}^{W}R_C$ represents the yaw rotation to the intermediate frame $\mathcal{C}$ and ${}^{C}R_B$ represents the effect of roll and pitch. The angular velocity of the robot is denoted by $\omega_{\mathcal{BW}}$, denoting the angular velocity of frame $\mathcal{B}$ in the frame $\mathcal{W}$, with components p, q, and r in the body frame. These values can be directly related to the derivatives of the roll, pitch, and yaw angles.

Each rotor has an angular speed ω_i and produces a force, F_i, and moment, M_i, according to

$$F_i = k_F \omega_i^2, \; M_i = k_M \omega_i^2.$$

In practice, the motor dynamics are relatively fast compared to the rigid body dynamics and the aerodynamics so for the controller development we assume they can be instantaneously achieved. Therefore the control input to the system can be written as $\mathbf{u}$ where u_1 is the net body force u_2, u_3, u_4 are the body moments which can be expressed according to the rotor speeds as

$$\mathbf{u} = \begin{bmatrix} k_F & k_F & k_F & k_F \\ 0 & k_F L & 0 & -k_F L \\ -k_F L & 0 & k_F L & 0 \\ k_M & -k_M & k_M & -k_M \end{bmatrix} \begin{bmatrix} \omega_1^2 \\ \omega_2^2 \\ \omega_3^2 \\ \omega_4^2 \end{bmatrix}, \quad (1)$$

where L is the distance from the axis of rotation of the propellers to the center of the quadrotor.

The position vector of the center of mass in the world frame is denoted by $\mathbf{r}$. The forces on the system are gravity, in the $-z_W$ direction, and the sum of the forces from each of the rotors, u_1, in the z_B direction. Newton's equations of motion governing the acceleration of the center of mass are

$$m\ddot{\mathbf{r}} = -mg\mathbf{z}_W + u_1 \mathbf{z}_B. \quad (2)$$

The angular acceleration determined by the Euler equations is

$$\dot{\omega}_{\mathcal{BW}} = \mathcal{I}^{-1} \left[-\omega_{\mathcal{BW}} \times \mathcal{I}\omega_{\mathcal{BW}} + \begin{bmatrix} u_2 \\ u_3 \\ u_4 \end{bmatrix} \right], \quad (3)$$

where $\mathcal{I}$ is the moment of inertia matrix referenced to the center of mass along the $x_B - y_B - z_B$ axes. The state of the system is given by the position and velocity of the center of mass and the orientation (locally parameterized by Euler angles) and the angular velocity:

$$\mathbf{x} = [x, y, z, \phi, \theta, \psi, \dot{x}, \dot{y}, \dot{z}, p, q, r]^T,$$

or without the parameterization by the the position and velocity of the center of mass and the rotation matrix ${}^{W}R_B$ and the angular velocity $\omega_{\mathcal{BW}}$.

III. Control

In [10] we presented a controller for following trajectories defined as the position and yaw angle as a function of time, $\mathbf{r}_T(t)$ and $\psi_T(t)$, respectively. We briefly outline that controller here for completeness. First we define the errors on position and velocity as

$$\mathbf{e}_p = \mathbf{r} - \mathbf{r}_T, \; \mathbf{e}_v = \dot{\mathbf{r}} - \dot{\mathbf{r}}_T.$$

Next we compute the desired force vector for the controller and the desired body frame z axis:

$$\mathbf{F}_{des} = -K_p \mathbf{e}_p - K_v \mathbf{e}_v + mg\mathbf{z}_W + m\ddot{\mathbf{r}}_T,$$

where K_p and K_v are positive definite gain matrices. Note that here we assume $\|\mathbf{F}_{des}\| \neq 0$. Next we project the desired force vector onto the actual body frame z axis in order to compute the desired force for the quadrotor and the first control input:

$$u_1 = \mathbf{F}_{des} \cdot \mathbf{z}_B. \quad (4)$$

To determine the other three inputs, we must consider the rotation errors. First, we observe that the desired z_B direction is along the desired thrust vector:

$$\mathbf{z}_{B,des} = \frac{\mathbf{F}_{des}}{\|\mathbf{F}_{des}\|}.$$

From the desired acceleration and a chosen yaw angle the total desired orientation can be found [10]. The orientation error is a function of the desired rotation matrix , R_{des}, and actual rotation matrix, ${}^{W}R_B$:

$$\mathbf{e}_R = \frac{1}{2}(R_{des}^T {}^{W}R_B - {}^{W}R_B^T R_{des})^\vee$$

where $^\vee$ represents the *vee map* which takes elements of $so(3)$ to $\mathbb{R}^3$. Note that the difference in Euler angles as used is [11] can be used as an approximation to this metric.

The angular velocity error is simply the difference between the actual and desired angular velocity in body frame coordinates:

$$\mathbf{e}_\omega = {}^{B}[\omega_{\mathcal{BW}}] - {}^{B}[\omega_{\mathcal{BW},T}].$$

Now the desired moments and the three remaining inputs are computed as follows:

$$[u_2, u_3, u_4]^T = -K_R \mathbf{e}_R - K_\omega \mathbf{e}_\omega, \quad (5)$$

where K_R and K_ω are diagonal gain matrices. This allows unique gains to be used for roll, pitch, and yaw angle tracking. Finally we compute the desired rotor speeds to achieve the desired $\mathbf{u}$ by inverting (1).

IV. Single Quadrotor Trajectory Generation

In this section we first describe the basic quadrotor trajectory generation method using Legendre polynomial functions incorporating obstacles into the formulation. Specifically, we solve the problem of generating smooth, safe trajectories through known 3-D environments satisfying specifications on intermediate waypoints.

A. Basic Method

Consider the problem of navigating a vehicle through n_w waypoints at specified times. A trivial trajectory that satisfies these constraints is one that interpolates between waypoints using straight lines. However this trajectory is inefficient because it has infinite curvature at the waypoints which requires the quadrotor to come to a stop at each waypoint. Our method generates an optimal trajectory that smoothly transitions through the waypoints at the given times. The optimization program to solve this problem while minimizing the integral of the k_rth derivative of position squared is shown below.

$$\begin{aligned} \min \quad & \int_{t_0}^{t_{n_w}} \left|\left| \tfrac{d^{k_r}\mathbf{r_T}}{dt^{k_r}} \right|\right|^2 dt \\ \text{s.t.} \quad & \mathbf{r}_T(t_w) = \mathbf{r}_w, \qquad w = 0, ..., n_w \\ & \tfrac{d^j x_T}{dt^j}|_{t=t_w} = 0 \text{ or free}, w = 0, n_w;\ j = 1, ..., k_r \\ & \tfrac{d^j y_T}{dt^j}|_{t=t_w} = 0 \text{ or free}, w = 0, n_w;\ j = 1, ..., k_r \\ & \tfrac{d^j z_T}{dt^j}|_{t=t_w} = 0 \text{ or free}, w = 0, n_w;\ j = 1, ..., k_r \end{aligned} \tag{6}$$

Here $\mathbf{r}_T = [x_T, y_T, z_T]^T$ and $\mathbf{r}_i = [x_i, y_i, z_i]^T$. We enforce continuity of the first k_r derivatives of $\mathbf{r}_T$ at t_1,...,t_{n_w-1}. Next we write the trajectories as piecewise polynomial functions of order n_p over n_w time intervals using polynomial basis functions $P_{pw}(t)$:

$$\mathbf{r}_T(t) = \begin{cases} \sum_{p=0}^{n_p} \mathbf{r}_{Tp1} P_{p1}(t) & t_0 \le t < t_1 \\ \sum_{p=0}^{n_p} \mathbf{r}_{Tp2} P_{p2}(t) & t_1 \le t < t_2 \\ \vdots & \\ \sum_{p=0}^{n_p} \mathbf{r}_{Tpn_w} P_{pn_w}(t) & t_{n_w-1} \le t \le t_{n_w} \end{cases} \tag{7}$$

This allows us formulate the problem as a quadratic program (or QP) by writing the constants $\mathbf{r}_{Tpw} = [x_{Tpw}, y_{Tpw}, z_{Tpw}]^T$ as a $3n_w n_p \times 1$ decision variable vector c:

$$\begin{aligned} \min \quad & c^T H c + f^T c \\ \text{s.t.} \quad & Ac \le b \end{aligned} \tag{8}$$

In our system, since the inputs u_2 and u_3 appear as functions of the fourth derivatives of the positions, we generate trajectories that minimize the integral of the square of the norm of the snap (the second derivative of acceleration, $k_r = 4$). The basis (7) allows us to go to higher order polynomials which allows us to satisfy such additional trajectory constraints as obstacle avoidance that are not explicitly specified by intermediate waypoints.

B. Choice of basis functions

Although this problem formulation is valid for any set of spanning polynomial basis functions, $P_{pw}(t)$, the choice does affect the numerical stability of the solver. A poor choice of basis functions can cause the matrix H in (8) to be ill-conditioned for large order polynomials. In order to diagonalize H and ensure that it is well-conditioned matrix we use Legendre polynomials as basis functions for the k_rth derivatives of our positions. Legendre polynomials are a spanning set of orthogonal polynomials on the interval from -1 to 1:

$$\int_{-1}^{1} \lambda_m(\tau)\lambda_n(\tau)d\tau = \frac{2}{2n+1}\delta_{nm}$$

where δ_{nm} is the Kronecker delta and τ is the non-dimensionalized time [4]. We then shift these Legendre polynomials to be orthogonal on the interval from t_{w-1} to t_w which we call $\lambda_{pw}(t)$. We use these shifted Legendre polynomials to represent the k_rth derivatives of the first $n_p - k_r$ basis functions for our position functions, $P_{pw}(t)$. These first $n_p - k_r$ polynomials must satisfy

$$\frac{d^{k_r} P_{pw}(t)}{dt^{k_r}} = \lambda_{pw}(t) \quad p = 1, ..., (n_p - k_r)$$

We define the last k_r polynomial basis functions as

$$P_{pw}(t) = (t - t_{w-1})^{p-n_p+k_r-1} \quad p = (n_p - k_r + 1), ..., n_p$$

Note these last k_r polynomial basis functions have no effect on the cost function because their k_rth derivatives are zero. In our work we take $k_r = 4$ and n_p is generally between 9 and 15.

C. Integer Constraints for Obstacle Avoidance

For collision avoidance we model the quadrotor as a rectangular prism oriented with the world frame with side lengths l_x, l_y, and l_z. These lengths are large enough so that the quadrotor can roll, pitch, and yaw to any angle and stay within the prism. We consider navigating this prism through an environment with n_o convex obstacles. Each convex obstacle o can be represented by a convex region in configuration space with $n_f(o)$ faces. For each face f the condition that the quadrotor's desired position at time t_k, $\mathbf{r}_T(t_k)$, be outside of obstacle o can be written as

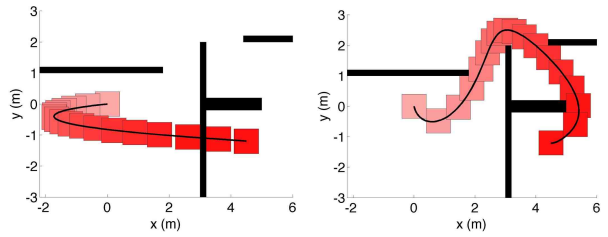
$$\mathbf{n}_{of} \cdot \mathbf{r}_T(t_k) \le s_{of}, \tag{9}$$

where $\mathbf{n}_{of}$ is the normal vector to face f of obstacle o in configuration space and s_{of} is a scalar that determines the location of the plane. If (9) is satisfied for *at least* one of the faces then the rectangular prism, and hence the quadrotor, is not in collision with the obstacle. The condition that the prism does not collide with an obstacle o at time t_k can be enforced with binary variables, b_{ofk}, as

$$\begin{aligned} \mathbf{n}_{of} \cdot \mathbf{r}_T(t_k) &\le s_{of} + M b_{ofk} \quad \forall f = 1, ..., n_f(o) \\ b_{ofk} &= 0 \text{ or } 1 \quad \forall f = 1, ..., n_f(o) \\ \sum_{f=1}^{n_f(o)} b_{ofk} &\le n_f(o) - 1 \end{aligned} \tag{10}$$

where M is a large positive number [14]. Note that if b_{ofk} is 1 then the inequality for face f is always satisfied. The last inequality in (10) requires that the non-collision constraint be satisfied for at least one face of the obstacle which implies that the prism does not collide with the obstacle. We can then introduce (10) into (8) for all n_o obstacles at n_k intermediate time steps between waypoints. The addition of the integer variables into the quadratic program causes this optimization problem to become a mixed-integer quadratic program (MIQP).

Note that this formulation is valid for any convex obstacle but we only consider rectangular obstacles in this paper for simplicity. This formulation is easily extended to moving obstacles by simply replacing $\mathbf{n}_{of}$ with $\mathbf{n}_{of}(t_k)$ and s_{of} with $s_{of}(t_k)$ in (10). Non-convex obstacles can also be efficiently modeled in the this framework as discussed in [16].



(a) Time-step overlap constraints are not enforced (b) Time-step overlap constraints are enforced

Fig. 3. Trajectories for a single quadrotor navigating an environment with four obstacles. Obstacles are the solid black boxes, the trajectory is shown as the black line, the position of the quadrotor at the n_k intermediate time steps for which collision checking is enforced is shown by the red boxes which grow darker with passing time. Note that for both of these trajectories $n_p = 15$ and $n_k = 16$

D. Discretization in Time

Equation (8) represents a continuous time optimization. We discretize time and write the collision constraints in (10) for n_k time points. However, collision constraints at n_k discrete times do not guarantee that the trajectory will be collision-free between the time steps. For a thin obstacle the optimal trajectory may cause the quadrotor to travel quickly through the obstacle such that the collision constraints are satisfied just before passing through the obstacle and just after as shown in Fig. 3(a). This problem can be fixed by requiring that the rectangular prism for which collision checking is enforced at time step k has a finite intersection with the corresponding prism for time step $k+1$:

$$\begin{aligned} |x_T(t_k) - x_T(t_k+1)| &\leq l_x \quad \forall k = 0, ..., n_k \\ |y_T(t_k) - y_T(t_k+1)| &\leq l_y \quad \forall k = 0, ..., n_k \\ |z_T(t_k) - z_T(t_k+1)| &\leq l_z \quad \forall k = 0, ..., n_k \end{aligned} \tag{11}$$

These additional *time-step overlap constraints* prevent the trajectory from passing through obstacles as shown in Fig. 3(b). Enforcing time-step overlap is equivalent to enforcing an average velocity constraint between time steps. Of course, enough time steps must be used so that a solution is feasible. Note that the trajectory may still cut corners due to the time discretization. We address this by appropriately inflating the size of the obstacles and prisms for which collision checking is enforced. After the trajectory is found we perform a collision check to ensure that the actual quadrotor shape does not intersect with any of the obstacles over the entire trajectory.

E. Temporal Scaling

As in [10] we can exploit temporal scaling to tradeoff between safety and aggressiveness. If we change the time to navigate the waypoints by a factor of α (e.g., $\alpha = 2$ allows the trajectory to be executed in twice as much time) the solution to the time-scaled problem is simply a time-scaled version of the original solution. Hence, we do not need to resolve the MIQP. As α is increased the plan takes longer to execute and becomes *safer*. As α goes to infinity all the derivatives of position and yaw angle as well as the angular velocity go to zero which leads, in the limit, to

$$\mathbf{u}(t) \rightarrow [mg, 0, 0, 0]^T,$$

in (4) and (5). By making α large enough we can satisfy any motion plan generated for a quadrotor with the assumption of small pitch and roll. Conversely, as α is decreased the trajectory takes less time to execute, the derivatives of position increase, and the trajectory becomes more aggressive leading to large excursions from the zero pitch and zero roll configuration.

V. Multiple Quadrotor Trajectory Generation

In this section we extend the method to include n_q heterogeneous quadrotors navigating in the same environment, often in close proximity, to designated goal positions, each with specified waypoints. This is done by solving a larger version of (8) where the decision variables are the trajectories coefficients of all n_q quadrotors. For collision avoidance constraints each quadrotor can be a different size as specified by unique values of l_x, l_y, l_z. We also consider heterogeneity terms with relative cost weighting and inter-quadrotor collision avoidance.

A. Relative Cost Weighting

A team of quadrotors navigating independently must resolve conflicts that lead to collisions and "share" the three-dimensional space. Thus they must modify their individual trajectories to navigate an environment and avoid each other. If all quadrotors are of the same type then it makes sense for them to share the burden of conflict resolution equally. However, for a team of heterogeneous vehicles it may be desirable to allow some quadrotors to follow relatively easier trajectories than others, or to prioritize quadrotors based on user preferences. This can be accomplished by weighting their costs accordingly. If quadrotor q has relative cost μ_q then the quadratic cost matrix, H_m, in the multi-quadrotor version of (8) can be written

$$H_m = \text{diag}(\mu_1 H_1, \mu_2 H_2, ..., \mu_{n_q} H_{n_q}) \tag{12}$$

Applying a larger weighting factor to a quadrotor lets it take a more direct path between its start and goal. Applying a smaller weighting factor forces a quadrotor to modify its trajectory to yield to other quadrotors with larger weighting factors. This ability is particularly valuable for a team of both agile and slow quadrotors as a trajectory for a slow, large quadrotor can be assigned a higher cost than the same trajectory for a smaller and more agile quadrotor. A large quadrotor requires better tracking accuracy than a small quadrotor to fly through the same narrow gap so it is also useful to assign higher costs for larger quadrotors in those situations.

B. Inter-Quadrotor Collision Avoidance

Quadrotors must stay a safe distance away from each other. We enforce this constraint at n_k intermediate time steps between waypoints which can be represented mathematically for quadrotors 1 and 2 by the following set of constraints:

$$\begin{aligned} \forall t_k: \quad & x_{1T}(t_k) - x_{2T}(t_k) \leq d_{x12} \\ \text{or} \quad & x_{2T}(t_k) - x_{1T}(t_k) \leq d_{x21} \\ \text{or} \quad & y_{1T}(t_k) - y_{2T}(t_k) \leq d_{y12} \\ \text{or} \quad & y_{2T}(t_k) - y_{1T}(t_k) \leq d_{y21} \\ \text{or} \quad & z_{1T}(t_k) - z_{2T}(t_k) \leq d_{z12} \\ \text{or} \quad & z_{2T}(t_k) - z_{1T}(t_k) \leq d_{z21} \end{aligned} \tag{13}$$

Here the d terms represent safety distances. For axially symmetric vehicles $d_{x12} = d_{x21} = d_{y12} = d_{y21}$. Experimentally we have found that quadrotors must avoid flying in the downwash of similar-sized or larger quadrotors because of a decrease in tracking performance and even instability in the worst cases. Larger quadrotors, however, can fly underneath smaller quadrotors. We have demonstrated that a larger quadrotor can even fly stably enough under a small quadrotor to serve as an aerial landing platform (see attached video). Therefore if quadrotor 1 and 2 are of the same type then $d_{z12} = d_{z21}$. However, if quadrotor 1 is much bigger than quadrotor 2 then quadrotor 2 must fly well below the larger quadrotor at some large distance d_{z12} while quadrotor 1 can fly much closer underneath quadrotor 2 represented by the smaller distance d_{z21}. The exact values of these safety distances can be found experimentally by measuring the tracking performance for different separation distances between quadrotor types. Finally, we incorporate constraints (13) between all n_q quadrotors in the same manner as in (10) into the multi-quadrotor version of (8).

C. Computational Complexity and Numerical Algorithm

Here we analyze the complexity of the MIQP generated by this formulation for a three-dimensional navigation problem formed by (8), (10), and (13). In this problem the number of continuous variables, n_c, is at most

$$n_c = 3 n_w n_p n_q. \tag{14}$$

Some continuous variables can be eliminated from the MIQP by removing the equality constraints. A strong factor that determines the computational time is the number of binary variables, n_b, that are introduced. The number of binary variables for a three-dimensional navigation problem is:

$$n_b = n_w n_k n_q \prod_{o=1}^{n_o} n_f(o) + n_w n_k \frac{n_q(n_q-1)}{2} 6 \tag{15}$$

The first term in (15) accounts for the obstacle avoidance constraints and the second term represents inter-quadrotor safety distance enforcement.

In this paper, we use a branch and bound solver [2] to solve the MIQP. At a worst case there are 2^{n_b} leaves of the tree to explore. Therefore, this is not a method that scales well for large number of robots but it can generate optimal trajectories for small teams (up to 4 quadrotors in this paper) and a few obstacles. Computational times for all scenarios presented in this paper are shown in Sec. VI-D. One advantage with this technique is that suboptimal, feasible solutions that guarantee safety and conflict resolution can be found very quickly (compare T_1 and T_{opt} in the Table I) if the available computational budget is low.

VI. Experimental Results

The experiments presented in this paper are conducted with Ascending Technologies Hummingbird quadrotors [1] as well as the kQuad65 and kQuad1000 quadrotors developed in-house which weigh 457, 65, and 962 grams and have blade tip to blade tip lengths of 55, 21, and 67 cm, respectively. We use a Vicon motion capture system [3] to estimate the position and velocity of the quadrotors and the onboard IMU to estimate the orientation and angular velocities. The software infrastructure is described in [12].

In previous work [10], the orientation error term, (5), was computed off-board the vehicle using the orientation as measured by the motion capture system. This off-board computation introduces a variable time delay in the control loop which is significant when using with multiple quadrotors. The time delay limits the performance of the attitude controller. We choose to instead use a stiff on-board linearized attitude controller as in [11] instead of the softer off-board nonlinear attitude controller as in [10].

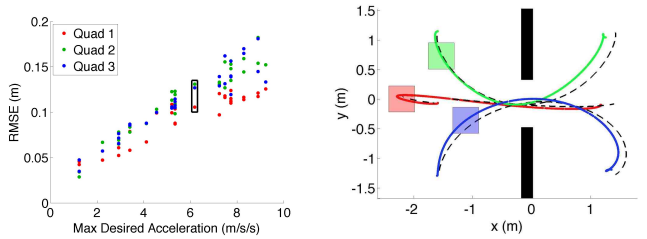
We solve all problems with the MIQP solver in the CPLEX software package [2]. Computational times for all scenarios presented in this paper are shown in Sec. VI-D.

A. Three Quadrotors in Plane with Obstacles

This experiment demonstrates planning for three vehicles in a planar scenario with obstacles. Three homogeneous Hummingbird quadrotors start on one side of a narrow gap and must pass through to goal positions on the opposite side. The trajectories were found using the method described in Sec. IV using 10th order polynomials and enforcing collision constraints at 11 intermediate time steps between the two waypoints ($n_p = 10$, $n_k = 11$, $n_w = 1$). The quadrotors were then commanded to follow these trajectories at various speeds for 30 trials with a hoop placed in the environment to represent the gap. Data and images for this experiment are shown in Figs. 4 and 5. Figure 4(a) shows the root-mean-square errors (RMSE) for each of these trials. While trajectories with larger acceleration, jerk, and snap do cause larger errors (as expected) the performance degrades quite gracefully. The data for a single run is presented in Figs. 4(b-d).

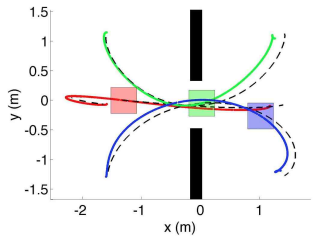
B. Two Heterogeneous Quadrotors through 3-D gap

The experiment demonstrates the navigation of a kQuad1000 (Quadrotor 1) and a Hummingbird (Quadrotor 2) from positions below a gap to positions on the opposite side of the room above the gap. This problem is formulated as a 3-D trajectory generation problem using 13th order polynomials and enforcing collision constraints at 9 intermediate time steps between the two waypoints ($n_p = 13$, $n_k = 9$, $n_w = 1$). For the problem formulation four three-dimensional rectangular prism shaped obstacles are used to create a single 3-D gap which the quadrotors must pass through to get to their goals. Data and images for these experiments are shown Figs. 6 and 7. Since the bigger quadrotor has a tighter tolerance to pass through the gap we choose to weight its cost function 10 times more than the

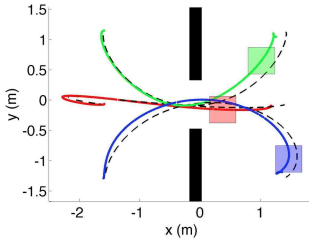


(a) RMSE for 30 trials at various speeds

(b) $t = 1.0s$



(c) $t = 1.5s$



(d) $t = 1.8s$

Fig. 4. Images (b-d) show data for a single run (the boxed data in (a)). The colored boxes represent the quadrotor positions at specified times during the experiments corresponding to the snapshots in Fig. 5. The colored lines represent the actual quadrotor trajectories for this run while the dotted black lines represent the desired trajectories.

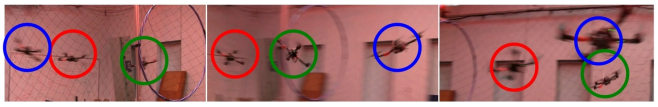
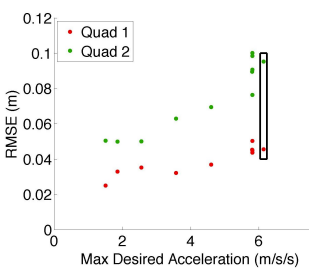


Fig. 5. Snapshots of the three quadrotor experiment in which the hoop represents the gap. See the attached video or http://tinyurl.com/multiquad.

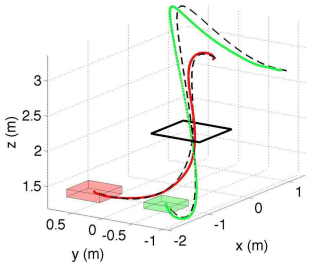
Hummingbird. This can be observed from the more indirect route taken by the quadrotor 2 in Fig. 6. Also, this can be observed by the larger error for quadrotor 2 since it is following a more difficult trajectory which requires larger velocities and accelerations. Finally, one should note that the larger quadrotor follows the smaller one up through the gap because it is allowed to fly underneath the smaller one but not vice versa as described in Sec. V-B.

C. Formation Reconfiguration with Four Quadrotors

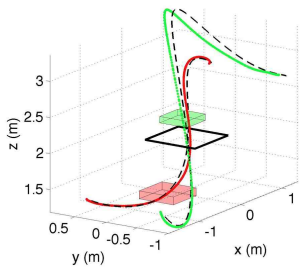
This experiment demonstrates reconfiguration for teams of four quadrotors. This problem is formulated as a 3-D trajectory generation problem using 9th order polynomials and enforcing collision constraints at 9 intermediate time steps between the two waypoints ($n_p = 9$, $n_k = 9$, $n_w = 1$). Trajectories are generated which transition quadrotors between arbitrary positions in a given three-dimensional formation or to a completely different formation smoothly and quickly. We present several reconfigurations in the attached video and a single reconfiguration within a line formation in Figs. 8 and 9. We ran the experiment with four Hummingbirds and a heterogeneous team consisting of two Hummingbirds, one kQuad65, and one kQuad1000. For the heterogeneous group we weight the cost of the kQuad65 10 times larger than the other quads because it is the least agile and can presently only follow moderately aggressive trajectories. Notice how the kQuad65 takes the most direct trajectory in 8(b). For the homogeneous experiment shown in



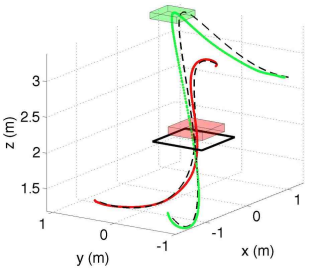
(a) RMSE for 11 trials at various speeds



(b) $t = 0.0s$



(c) $t = 1.9s$



(d) $t = 2.3s$

Fig. 6. Images (b-d) show data for a single run (the boxed data in (a)). The colored boxes represent the quadrotor positions at specified times during the experiment corresponding to the snapshots in Fig. 7. The colored lines represent the actual quadrotor trajectories for this run while the dotted black lines represent the desired trajectories.

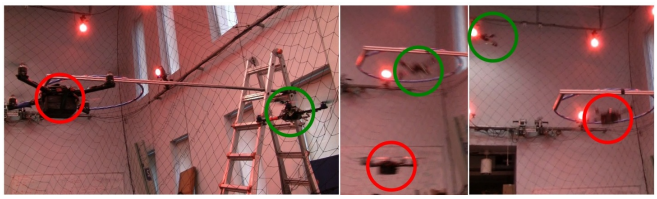


Fig. 7. Snapshots of an experiment with the kQuad1000 (quadrotor 1, red) and the AscTec Hummingbird (quadrotor 2, green) in which the hoop represents the horizontal gap. See the attached video or http://tinyurl.com/multiquad.

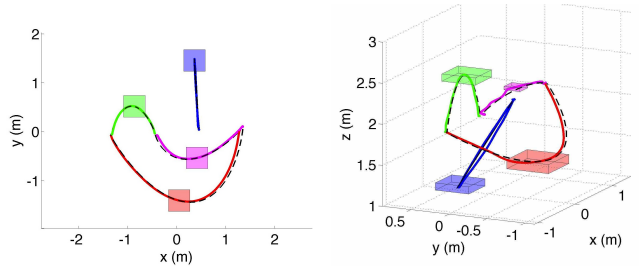
Fig. 8(a) the quadrotors stay in the same plane because they are not allowed to fly underneath each other as described in Sec. V-B but in the heterogeneous experiment shown in Fig. 8(b) the optimal solution contains z components since larger quadrotors are allowed to fly under smaller ones.

D. Solver Details

We present problem details and computational times for each of the MIQPs solved in this paper in Table I. All computation times are listed for a MacBook Pro laptop with a 2.66 GHz Intel Core 2 Duo processor using the CPLEX MIQP solver [2]. Note that while certain problems take a long time to find the optimal solution and prove optimality, a first solution is always found in less than a second. The solver can be stopped any time after the first feasible is found and return a sub-optimal solution.

VII. Concluding Remarks

We presented an algorithm for generating optimal trajectories for multiple heterogeneous quadrotors in environments with obstacles. This method can enforce constraints on positions, velocities, accelerations, jerks and inputs and allows



(a) Four Hummingbirds - Top View (b) kQuad1000 (red), kQuad65 (magenta), and two Hummingbirds (green and blue) - Perspective View

Fig. 8. Trajectories for formation reconfigurations for homogeneous (a) and heterogeneous (b) quadrotor teams. The colored boxes represent the quadrotor positions at an intermediate time during the trajectories. The colored lines represent the actual quadrotor trajectories while the dotted black lines represent the desired trajectories.

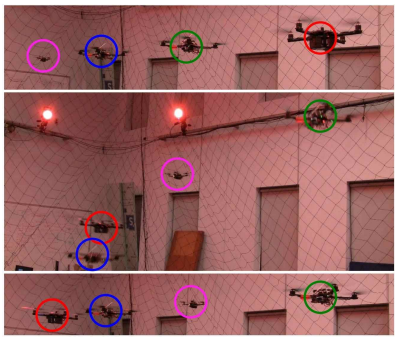


Fig. 9. Snapshots of a four quadrotor reconfiguration within a line formation at the beginning (top), an intermediate time (middle), and the final time (bottom). The four quadrotors used are the kQuad1000 (red), the kQuad65 (magenta), and two Hummingbirds (green and blue). See the attached video or http://tinyurl.com/multiquad.

for different sizes, capabilities, and varying dynamic effects between different quadrotors. Collision avoidance is enforced using integer constraints. The trajectories are optimal in the sense that they minimize cost functionals that are derived from the square of the norm of the snap (the fourth derivative of position). The time scaling property of this approach allows trajectories to be slowed down to be made *safer*.

This method is not without its limitations. We acknowledge that this is a centralized approach that requires knowledge of the start and goal positions for all agents. The computational complexity of the MIQP limits the application of this method to small teams with a small number of obstacles. Complexity also increases with the number of time steps for which collision avoidance is enforced so there is a tradeoff between plan fidelity and planning time. However, as shown in Table I suboptimal solutions are often available orders of magnitude faster than the optimal solution. Nonetheless, large teams and more complex environments will require a different planning paradigm. A subject of future work is the effective combination of this method with search-based planning algorithms in order to capitalize on the strengths of both approaches.

Fig.	n_q	n_p	n_k	n_b	T_1 (s)	T_{opt} (s)
3(b)	1	15	16	208	0.42	35
6	2	13	9	270	0.62	1230
4	3	10	11	300	0.21	553
8(a)	4	9	9	324	0.11	39
8(b)	4	9	9	324	0.45	540

TABLE I

T_1 IS THE TIME TO FIND THE FIRST FEASIBLE SOLUTION AND T_{opt} IS THE TIME TO FIND THE OPTIMAL SOLUTION AND PROVE ITS OPTIMALITY.

VIII. Acknowledgements

The kQuad65 and kQuad1000 were developed by Alex Kushleyev and Max Likhachev. We gratefully acknowledge the support of DARPA CSSP Grant N10AP20011, ARO Grant W911NF-05-1-0219, ONR Grants N00014-07-1-0829, N00014-08-1-0696 and N00014-09-1-1051, and ARL Grant W911NF-08-2-0004.

References

[1] Ascending Technologies, GmbH. http://www.asctec.de.
[2] IBM ILOG CPLEX V12.1: Users manual for CPLEX, International Business Machines Corporation, 2009.
[3] Vicon Motion Systems, Inc. http://www.vicon.com.
[4] M. Abramowitz and I. Stegun. *Handbook of Mathematical Functions*. Dover, first edition, 1964.
[5] J. H. Gillula, H. Huang, M. P. Vitus, and C. J. Tomlin. Design of guaranteed safe maneuvers using reachable sets: Autonomous quadrotor aerobatics in theory and practice. In *Proc. of the IEEE Int. Conf. on Robotics and Automation*, pages 1649–1654, Anchorage, AK, May 2010.
[6] D. Gurdan, J. Stumpf, M. Achtelik, K. Doth, G. Hirzinger, and D. Rus. Energy-efficient autonomous four-rotor flying robot controlled at 1 khz. In *Proc. of the IEEE Int. Conf. on Robotics and Automation*, Roma, Italy, Apr. 2007.
[7] R. He, A. Bachrach, M. Achtelik, A. Geramifard, D. Gurdan, S. Prentice, J. Stumpf, and N. Roy. On the design and use of a micro air vehicle to track and avoid adversaries. *The Int. Journal of Robotics Research*, 29:529–546, 2010.
[8] M. Likhachev, G. Gordon, and S. Thrun. ARA*: Anytime A* with provable bounds on sub-optimality. *Advances in Neural Information Processing Systems*, 16, 2003.
[9] S. Lupashin, A. Schollig, M. Sherback, and R. D'Andrea. A simple learning strategy for high-speed quadrocopter multi-flips. In *Proc. of the IEEE Int. Conf. on Robotics and Automation*, pages 1642–1648, Anchorage, AK, May 2010.
[10] D. Mellinger and V. Kumar. Minimum snap trajectory generation and control for quadrotors. In *Proc. of the IEEE Int. Conf. on Robotics and Automation*, May 2011.
[11] D. Mellinger, N. Michael, and V. Kumar. Trajectory generation and control for precise aggressive maneuvers. In *Int. Symposium on Experimental Robotics*, Dec. 2010.
[12] N. Michael, D. Mellinger, Q. Lindsey, and V. Kumar. The grasp multiple micro uav testbed. *IEEE Robotics and Automation Magazine*, Sept. 2010.
[13] A. Richards, J. How, T. Schouwenaars, and E. Feron. Plume avoidance maneuver planning using mixed integer linear programming. In *AIAA Guidance, Navigation and Control Conference and Exhibit*, 2001.
[14] T. Schouwenaars, B. DeMoor, E. Feron, and J. How. Mixed integer programming for multi-vehicle path planning. In *European Control Conference*, pages 2603–2608, 2001.
[15] T. Schouwenaars, J. How, and E. Feron. Receding horizon path planning with implicit safety guarantees. In *American Control Conference*, pages 5576–5581, 2004.
[16] T. Schouwenaars, A. Stubbs, J. Paduano, and E. Feron. Multi-vehicle path planning for non-line of sight communication. In *American Control Conference*, 2006.
[17] R. Tedrake. LQR-Trees: Feedback motion planning on sparse randomized trees. In *Proc. of Robotics: Science and Systems*, Seattle, WA, June 2009.