Towards A Swarm of Agile Micro Quadrotors

Alex Kushleyev, Daniel Mellinger, Vijay Kumar
GRASP Lab, University of Pennsylvania

***Abstract*—We describe a prototype 73 gram, 21 cm diameter micro quadrotor with onboard attitude estimation and control that operates autonomously with an external localization system. We argue that the reduction in size leads to agility and the ability to operate in tight formations and provide experimental arguments in support of this claim. The robot is shown to be capable of 1850°/sec roll and pitch, performs a 360° flip in 0.4 seconds and exhibits a lateral step response of 1 body length in 1 second. We describe the architecture and algorithms to coordinate a team of quadrotors, organize them into groups and fly through known three-dimensional environments. We provide experimental results for a team of 20 micro quadrotors.**

I. Introduction

The last decade has seen rapid progress in micro aerial robots, autonomous aerial vehicles that are smaller than 1 meter in scale and 1 kg or less in mass. Winged aircrafts can range from fixed-wing vehicles [14] to flapping-wing vehicles [6], the latter mostly inspired by insect flight. Rotor crafts, including helicopters, coaxial rotor crafts [9], ducted fans[22], quadrotors [10] and hexarotors, have proved to be more mature [15] with quadrotors being the most commonly used aerial platform in robotics research labs. In this class, the Hummingbird quadrotor sold by Ascending Technologies, GmbH [2], with a tip-to-tip wingspan of 55 cm, a height of 8 cm, mass of about 500 grams including a Lithium Polymer battery and consuming about 75 Watts is a remarkably capable and robust platform as shown in [16, 17].

Of course micro aerial robots have a fundamental payload limitation that is difficult to overcome in many practical applications. However larger payloads can be manipulated and transported by multiple UAVs either using grippers or cables [20]. Applications such as surveillance or search and rescue that require coverage of large areas or imagery from multiple sensors can be addressed by coordinating multiple UAVs, each with different sensors.

Our interest in this paper is scaling down the quadrotor platform to develop a truly small micro UAV. The most important and obvious benefit of scaling down in size is the ability of the quadrotor to operate in tightly constrained environments in tight formations. While the payload capacity of the quadrotor falls dramatically, it is possible to deploy multiple quadrotors that cooperate to overcome this limitation. Again, the small size benefits us because smaller vehicles can operate in closer proximity than large vehicles. Another interesting benefit of scaling down is agility. As argued later and illustrated with experimental results, smaller quadrotors exhibit higher accelerations allowing more rapid adaptation to disturbances and higher stability.

Fig. 1. A formation of 20 micro quadrotors in flight. See video at http://youtu.be/50Fdi7712KQ

II. Agility of Micro Quadrotors

It is useful to develop a simple physics model to analyze a quadrotor's ability to produce linear and angular accelerations from a hover state. If the characteristic length is L, the rotor radius R scales linearly with L. The mass scales as L^3 and the moments of inertia as L^5. On the other hand the lift or thrust, F, and drag, D, from the rotors scale with the cross-sectional area and the square of the blade-tip velocity, v. If the angular speed of the blades is defined by $\omega = \frac{v}{L}$, $F \sim \omega^2 L^4$ and $D \sim \omega^2 L^4$. The linear acceleration a scales as $a \sim \frac{\omega^2 L^4}{L^3} = \omega^2 L$. Thrusts from the rotors produce a moment with a moment arm L. Thus the angular acceleration $\alpha \sim \frac{\omega^2 L^5}{L^5} = \omega^2$.

The rotor speed, ω also scales with length since smaller motors produce less torque which limits their peak speed because of the drag resistance that also scales the same way as lift. There are two commonly accepted approaches to study scaling in aerial vehicles [28]. Mach scaling is used for compressible flows and essentially assumes that the tip velocities are constant leading to $\omega \sim \frac{1}{R}$. Froude scaling is used for incompressible flows and assumes that for similar aircraft configurations, the Froude number, $\frac{v^2}{Lg}$, is constant. Here g is the acceleration due to gravity. This yields $\omega \sim \frac{1}{\sqrt{R}}$. However, neither Froude or Mach number similitudes take motor characteristics nor battery properties into account. While motor torque increases with length, the operating speed for the rotors is determined by matching the torque-speed characteristics of the motor to the drag versus speed characteristics of the propellors. Further, the motor torque depends on the ability of the battery to source the required current. All these variables are tightly coupled for smaller designs since there are fewer choices available at smaller length scales. Finally, the assumption that propeller blades are rigid may be wrong and the performance of the blades can be very different at smaller scales, the quadratic scaling of the lift with speed may

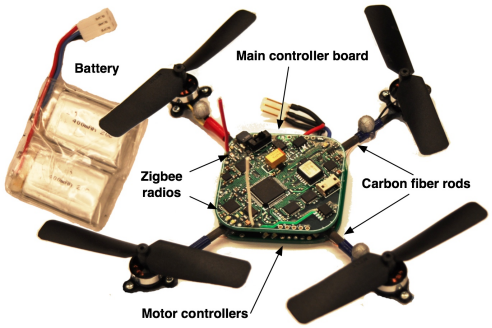
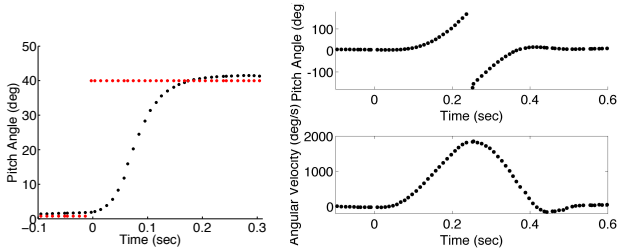


Fig. 2. A prototype micro quadrotor.



(a) Pitch angle step input response

(b) Data for the flipping maneuver

Fig. 3. Attitude controller performance data

not be accurate. Nevertheless these two cases are meaningful since they provide some insight into the physics underlying the maneuverability of the craft.

Froude scaling suggests that the acceleration is independent of length while the angular acceleration $\alpha \sim L^{-1}$. On the other hand, Mach scaling leads to the conclusion that $a \sim L$ while $\alpha \sim L^{-2}$. Since quadrotors must rotate (exhibit angular accelerations) in order to translate, smaller quadrotors are much more agile.

There are two design points that are illustrative of the quadrotor configuration. The Pelican quadrotor from Ascending Technologies [2] equipped with sensors (approx. 2 kg gross weight, 0.75 m diameter, and 5400 rpm nominal rotor speed at hover), consumes approximately 400 W of power [25]. The Hummingbird quadrotor from Ascending Technologies (500 grams gross weight, approximately 0.5 m diameter, and 5000 rpm nominal rotor speed at hover) without additional sensors consumes about 75 W. In this paper, we outline a design for a quadrotor which is approximately 40% of the size of the Hummingbird, 15% of its mass, and consuming approximately 20% of the power for hovering.

III. The Micro Quadrotor

A. The Vehicle

The prototype quadrotor is shown in Figure 2. Its booms are made of carbon fiber rods which are sandwiched between a custom motor controller board on the bottom and the main controller board on the top. To produce lift the vehicle uses four fixed-pitch propellers with diameters of 8 cm. The vehicle propeller-tip-to-propeller-tip distance is 21 cm and its weight without a battery is 50 grams. The hover time is approximately 11 minutes with a 2-cell 400 mAh Li-Po battery that weighs 23 grams.

B. Electronics

Despite its small size this vehicle contains a full suite of onboard sensors. An ARM Cortex-M3 processor, running at 72 MHz, serves as the main processor. The vehicle contains a 3-axis magnetometer, a 3-axis accelerometer, a 2-axis 2000 deg/sec rate gyro for the roll and pitch axes, and a single-axis 500 deg/sec rate gyro for the yaw axis. The vehicle also contains a barometer which can be used to sense a change in altitude. For communication the vehicle contains two Zigbee transceivers that can operate at either 900 MHz or 2.4 GHz.

C. Software Infrastructure

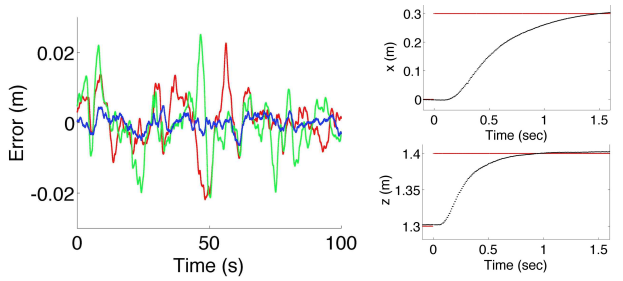
The work in this paper uses a Vicon motion capture system [5] to sense the position of each vehicle at 100 Hz. This data is streamed over a gigabit ethernet network to a desktop base station. High-level control and planning is done in MATLAB on the base station which sends commands to each quadrotor at 100 Hz. The software for controlling a large team of quadrotors is described later in Sec. V (see Fig. 7). Low-level estimation and control loops run on the onboard microprocessor at a rate of 600 Hz.

Each quadrotor has two independent radio transceivers, operating at 900 MHz and 2.4 GHz. The base station sends, via custom radio modules, the desired commands, containing orientation, thrust, angular rates and attitude controller gains to the individual quadrotors. The onboard rate gyros and accelerometer are used to estimate the orientation and angular velocity of the craft. The main microprocessor runs the attitude controller described in Sec. IV and sends the desired propeller speeds to each of the four motor controllers at full rate (600Hz).

D. Performance

Some performance data for the onboard attitude controller in Fig. 3. The small moments of inertia of the vehicle enable the vehicle to create large angular accelerations. As shown in Fig. 4(a) the attiude control is designed to be approximately critically damped with a settling time of less than 0.2 seconds. Note that this is twice as fast as the settling time for the attitude controller for the AscTec Hummingbird reported in [18]. Data for a flip is presented 3(b). Here the vehicle completes a complete flip about its y axis in about 0.4 seconds and reaches a maximum angular velocity of 1850 deg/sec.

The position controller described in Sec. IV uses the roll and pitch angles to control the x and y position of the vehicle. For this reason, a stiff attitude controller is a required for stiff position control. Response to step inputs in the lateral and vertical directions are shown in Fig. 4(b). For the hovering performance data shown in Fig. 4 the standard deviations of the error for x and y are about 0.75 cm and about 0.2 cm for z.



(a) Position error

(b) Position step input response

Fig. 4. The red, green, and blue lines in (a) represent the x, y, and z errors while hovering. (b) shows the step response for the position controller in x (top) and z (bottom).

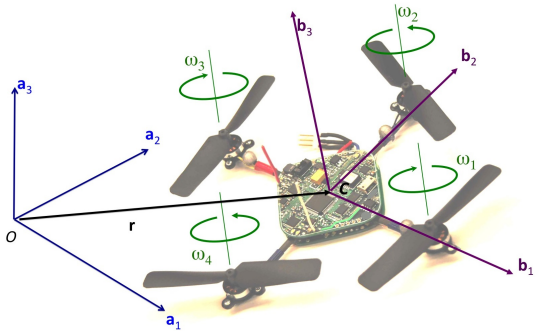


Fig. 5. The reference frames and propeller numbering convention.

IV. Dynamics and Control

The dynamic model and control for the micro quadrotor is based on the approach in [17]. As shown in Figure 5, we consider a body-fixed frame $\mathcal{B}$ aligned with the principal axes of the quadrotor (unit vectors $\mathbf{b}_i$) and an inertial frame $\mathcal{A}$ with unit vectors $\mathbf{a}_i$. $\mathcal{B}$ is described in $\mathcal{A}$ by a position vector $\mathbf{r}$ to the center of mass C and a rotation matrix R. In order to avoid singularities associated with parameterization, we use the full rotation matrix to describe orientations. The angular velocity of the quadrotor in the body frame, ω, is given by $\hat{\omega} = R^T\dot{R}$, where $\hat{}$ denotes the skew-symmetric matrix form of the vector.

As shown in Fig. 5, the four rotors are numbered 1-4, with odd numbered rotors having a pitch that is opposite to the even numbered rotors. The angular speed of the rotor is ω_i. The resulting lift, F_i, and the reaction moment, M_i, are given by:

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

where the constants k_F and k_M are empirically determined. For our micro quadrotor, the motor dynamics have a time constant less than 10 msec and are much faster than the time scale of rigid body dynamics and aerodynamics. Thus we neglect the dynamics and assume F_i and M_i can be instantaneously changed. Therefore the control input to the system, $\mathbf{u}$, consists of the net thrust in the $\mathbf{b}_3$ direction, $u_1 = \Sigma_{i=1}^4 F_i$, and the moments in $\mathcal{B}$, $[u_2, u_3, u_4]^T$, given by:

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

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

The Newton-Euler equations of motion are given by:

$$m\ddot{\mathbf{r}} = -mg\mathbf{a}_3 + u_1\mathbf{b}_3 \quad (2)$$

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

where $\mathcal{I}$ is the moment of inertia matrix along $\mathbf{b}_i$.

We specify the desired trajectory using a time-parameterized position vector and yaw angle. Given a trajectory, $\sigma(t) : [0, t_f] \to \mathbb{R}^3 \times SO(2)$, the controller derives the input u_1 based on position and velocity errors:

$$u_1 = (-K_p\mathbf{e}_p - K_v\mathbf{e}_v + mg\mathbf{a}_3) \cdot \mathbf{b}_3 \quad (4)$$

where $\mathbf{e}_p = \mathbf{r} - \mathbf{r}_T$ and $\mathbf{e}_v = \dot{\mathbf{r}} - \dot{\mathbf{r}}_T$. The other three inputs are determined by computing the desired rotation matrix. We want to align the thrust vector $u_1\mathbf{b}_3$ with $(-K_p\mathbf{e}_p - K_v\mathbf{e}_v + mg\mathbf{a}_3)$ in (4). Second, we want the yaw angle to follow the specified yaw $\psi_T(t)$. From these two pieces of information we can compute R_{des} and the error in rotation according to:

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

where $^\vee$ represents the *vee map* which takes elements of $so(3)$ to $\mathbb{R}^3$. The desired angular velocity is computed by differentiating the expression for R and the desired moments can be expressed as a function of the orientation error, $\mathbf{e}_R$, and the angular velocity error, $\mathbf{e}_\omega$:

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

where K_R and K_ω are diagonal gain matrices. Finally we compute the desired rotor speeds to achieve the desired $\mathbf{u}$ by inverting (1).

V. Control and Planning for Groups

A. Architecture

We are primarily interested in the challenge of coordinating a large team of quadrotors. To manage the complexity that results from growth of the state space dimensionality and limit the combinatorial explosion arising from interactions between labeled vehicles, we consider a team architecture in which the team is organized into labeled groups, each with labeled vehicles. Formally, we can define a *group* of agents as a collection of agents which work simultaneously to complete a single task. Two or more groups act in a *team* to complete a task which requires completing multiple parallel subtasks [7]. We assume that vehicles within a group can communicate at high data rates with low latencies while the communication requirements for coordination across groups are much less stringent. Most importantly, vehicles within a group are labeled. The small group size allows us to design controllers and planners that provide global guarantees on

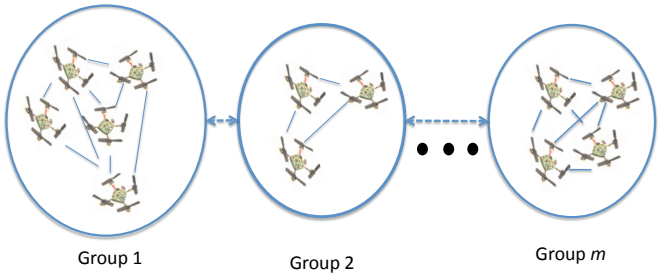


Fig. 6. The team of quadrotors is organized into m groups. While vehicles within the group are tightly coordinated and centralized control and planning is possible, the inter-group coordination need not be centralized.

shapes, communication topology, and relative positions of individual, agile robots.

Our approach is in contrast to truly decentralized approaches which are necessary in swarms with hundreds and thousands of agents [21]. While models of leaderless aggregation and swarming with aerial robots are discussed in the robotics community [11, 26, 19], here the challenge of enumerating labelled interactions between robots is circumvented by controlling such aggregate descriptors of formation as statistical distributions. These methods cannot provide guarantees on shape or topology. Reciprocal collision avoidance algorithms [27] have the potential to navigate robots to goal destinations but no guarantees are available for transient performance and no proof of convergence is available.

On the other hand, the problem of designing decentralized controllers for trajectory tracking for three dimensional rigid structures is now fairly well understood[12, 13, 8], although few experimental results are available for aerial robots. Our framework allows the maintenance of such rigid structures in groups.

B. Formation Flight

Flying in formation reduces the complexity of generating trajectories for a large team of vehicles to generating a trajectory for a single entity. If the controllers are well-designed, there is no need to explicitly incorporate collision avoidance between vehicles. The position error for quadrotor q at time t can be written as

$$\mathbf{e}_{pq}(t) = \mathbf{e}_f(t) + \mathbf{e}_{lq}(t) \tag{6}$$

where $\mathbf{e}_f(t)$ is the *formation error* describing the error of position of the group from the prescribed trajectory, and $\mathbf{e}_{lq}(t)$ is the local error of quadrotor q within the formation of the group. As we will show in Sec. VI the local error is typically quite small even for aggressive trajectories even though the formation error can be quite large.

A major disadvantage of formation flight is that the rigid formation can only fit through large gaps. This can be addressed by changing the shape of the formation of the team or dividing the team into smaller groups, allowing each group to negotiate the gap independently.

C. Time-Separated Trajectory Following

Another way to reduce the complexity of the trajectory generation problem is to require all vehicles to follow the same team trajectory but be separated by some time increment. Here we let the trajectory for quadrotor q be defined as

$$\mathbf{r}_{Tq}(t) = \mathbf{r}_{TT}(t + \Delta t_q) \tag{7}$$

where $\mathbf{r}_{TT}$ is the team trajectory and Δt_q is the time shift for quadrotor q from some common clock, t. If the team trajectory does not intersect or come within an unsafe distance of itself then vehicles simply need to follow each other at a safe time separation. Large numbers of vehicles can follow team trajectories that intersect themselves if the time separations, Δt_q, are chosen so that no two vehicles are at any of the intersection points at the same time. An experiment for an intersecting team trajectory is shown in Sec. VI.

D. Trajectory Generation with MIQPs

Here we describe a method for generating smooth, safe trajectories through known 3-D environments satisfying specifications on intermediate waypoints for multiple vehicles. Integer constraints are used to enforce collision constraints with obstacles and other vehicles and also to optimally assign goal positions. This method draws from the extensive literature on mixed-integer linear programs (MILPs) and their application to trajectory planning from Schouwenaars et al. [23, 24].

1) Basic Method: As described in [17] an optimization program can be used to generate trajectories that smoothly transition through n_w desired waypoints at specified times, t_w. The optimization program to solve this problem while minimizing the integral of the k_rth derivative of position squared for n_q quadrotors is shown below.

$$\begin{aligned}
&\min \textstyle\sum_{q=1}^{n_q} \int_{t_0}^{t_{n_w}} \left|\left| \frac{d^{k_r} \mathbf{r_{Tq}}}{dt^{k_r}} \right|\right|^2 dt \qquad (8)\\
&\text{s.t.} \quad \mathbf{r}_{Tq}(t_w) = \mathbf{r}_{wq}, \qquad w = 0, ..., n_w;\ \forall q\\
&\quad \tfrac{d^j x_{Tq}}{dt^j}|_{t=t_w} = 0 \text{ or free},\ w = 0, n_w;\ j = 1, ..., k_r;\ \forall q\\
&\quad \tfrac{d^j y_{Tq}}{dt^j}|_{t=t_w} = 0 \text{ or free},\ w = 0, n_w;\ j = 1, ..., k_r;\ \forall q\\
&\quad \tfrac{d^j z_{Tq}}{dt^j}|_{t=t_w} = 0 \text{ or free},\ w = 0, n_w;\ j = 1, ..., k_r;\ \forall q
\end{aligned}$$

Here $\mathbf{r}_{Tq} = [x_{Tq}, y_{Tq}, z_{Tq}]$ represents the trajectory for quadrotor q and $\mathbf{r}_{wq}$ represents the desired waypoints for quadrotor q. We enforce continuity of the first k_r derivatives of $\mathbf{r}_{Tq}$ at t_1,...,t_{n_w-1}. As shown in [17] writing the trajectories as piecewise polynomial functions allows [8] to be written as a quadratic program (or QP) in which the decision variables are the coefficients of the polynomials.

For quadrotors, 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, $k_r = 4$). Large order polynomials are used to satisfy such additional trajectory constraints as obstacle avoidance that are not explicitly specified by intermediate waypoints.

2) Integer Constraints for Collision Avoidance: For collision avoidance we model the quadrotors as a rectangular prisms 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, $\mathbf{r}_{Tq}(t_k)$, be outside of obstacle o can be written as

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

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 quadrotor q does not collide with an obstacle o at time t_k can be enforced with binary variables, b_{qofk}, as

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

where M is a large positive number [23]. Note that if b_{qofk} 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_q quadrotors 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).

3) Inter-Quadrotor Collision Avoidance: When transitioning between waypoints 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:

$$\begin{aligned} \forall t_k : \quad & x_{T1}(t_k) - x_{T2}(t_k) \leq d_{x12} \\ \text{or} \quad & x_{T2}(t_k) - x_{T1}(t_k) \leq d_{x21} \\ \text{or} \quad & y_{T1}(t_k) - y_{T2}(t_k) \leq d_{y12} \\ \text{or} \quad & y_{T2}(t_k) - y_{T1}(t_k) \leq d_{y21} \end{aligned} \tag{11}$$

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 each other's downwash because of a decrease in tracking performance and even instability in the worst cases. Therefore we do not allow vehicles to fly underneath each other here. Finally, we incorporate constraints (11) between all n_q quadrotors in the same manner as in (10) into (8).

4) Integer Constraints for Optimal Goal Assignment: In many cases one might not care that a certain quadrotor goes to a certain goal but rather that any vehicle does. Here we describe a method for using integer constraints to find the optimal goal assignments for the vehicles. This results in a lower total cost compared to fixed-goal assignment and often a faster planning time because there are more degrees of freedom in the optimization problem. For each quadrotor q and goal g we introduce the integer constraints:

$$\begin{aligned} x_{Tq}(t_{n_w}) &\leq x_g + M\beta_{qg} \\ x_{Tq}(t_{n_w}) &\geq x_g - M\beta_{qg} \\ y_{Tq}(t_{n_w}) &\leq y_g + M\beta_{qg} \\ y_{Tq}(t_{n_w}) &\geq y_g - M\beta_{qg} \\ z_{Tq}(t_{n_w}) &\leq z_g + M\beta_{qg} \\ z_{Tq}(t_{n_w}) &\geq z_g - M\beta_{qg} \end{aligned} \tag{12}$$

Here β_{qg} is a binary variable used to enforce the optimal goal assignment. If β_{qg} is 0 then quadrotor q must be at goal g at t_{n_w}. If β_{qg} is 1 then these constraints are satisfied for any final position of quadrotor q. In order to guaruntee that at least n_g quadrotors reach the desired goals we introduce the following constraint.

$$\sum_{q=1}^{n_q} \sum_{g=1}^{n_g} \beta_{qg} \leq n_g n_q - n_g \tag{13}$$

Note that this approach can be easily adapted if there are more quadrotors than goals or vice versa.

5) Relaxations for Large Teams: The solving time of the MIQP grows exponentially with the number of binary variables that are introduced into the MIQP. Therefore, the direct use of this method does not scale well for large teams. Here we present two relaxations that enable this approach to be used for large teams of vehicles.

a) Planning for Groups within a Team: A large team of vehicles can be divided into smaller groups. We can then use the MIQP method to generate trajectories to transition groups of vehicles to group goal locations. This reduces the complexity of the MIQP because instead of planning trajectories for all n_q vehicles we simply plan trajectories for the groups. Of course we are making a sacrifice here by not allowing the quadrotors to have the flexibility to move independently.

b) Planning for Sub-Regions: In many cases the environment can be partitioned into n_r convex sub-regions where each sub-region contains the same number of quadrotor start and goal positions. After partitioning the environment the MIQP trajectory generation method can be used for the vehicles inside each region. Here we require quadrotors to stay inside their own regions using linear constraints on the positions of the vehicles. This approach guarantees collision-free trajectories and allows quadrotors the flexibility to move independently. We are gaining tractability at the expense of optimality since the true optimal solution might actually require quadrotors to cross region boundaries while this relaxed version does not. Also, it is possible that no feasible trajectories exist inside a sub-region but feasible trajectories do exist which cross region boundaries. Nonetheless, this approach works well in many scenarios and we show its application to formation transitions for teams of 16 vehicles in Sec. VI.

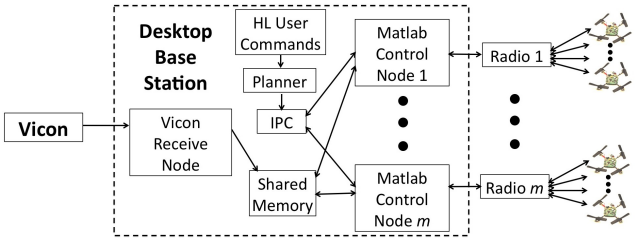
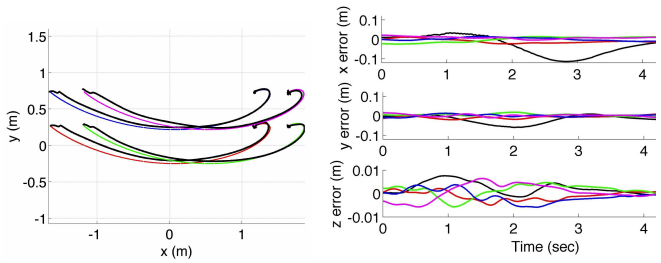
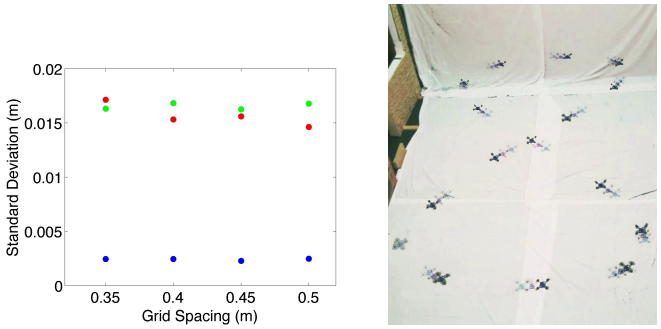


Fig. 7. Software Infrastructure



(a) Top view (b) Error time histories

Fig. 8. Formation following for a 4 quadrotor trajectory. In (a) the colored lines represent the desired trajecotories for each of the four vehicles and the black lines represent the actual trajecotries. The errors are shown in (b). Here the black line represents the formation error, $\mathbf{e}_f(t)$, from the desired trajectory and the colored lines represent the local errors, $\mathbf{e}_{li}(t)$, for each quadrotor.



(a) Error Data (b) Time-Separated Trajectory Following

Fig. 9. Part (a) shows the Average Standard Deviation for x, y, and z (shown in red, green and blue respectively) for 20 quadrotors in a grid formation. In (b) we show 16 quadrotors following a figure eight pattern. See the video at http://youtu.be/50Fdi7712KQ

VI. Experimental Results

A. Software Infrastructure for Groups

Our architecture (Sec. V-A) is important for a very practical reason. For a large team of quadrotors it is impossible to run a single loop that can receive all the Vicon data, compute the commands, and communicate with each quadrotor at a fast enough rate. As shown in Fig. 7, each group is controlled by a dedicated software node, running in an independent thread. These control nodes receive vehicle pose data from a special Vicon node via shared memory. The node connects to the Vicon tracking system, receives marker positions for each subject, performs a 6D pose fit to the marker data and additional processing for velocity estimation. Finally, the processed pose estimates are published to the shared memory using the Boost C++ library [3]. Shared memory is the fastest method of inter-process communication, which ensures the lowest latency of the time-critical data.

The control nodes, implemented in Matlab, read the pose data directly from shared memory and compute the commanded orientation and net thrusts for several quadrotors based on the controller described in IV. For non-time-critical data sharing we use Inter Process Communication (IPC) [4]. For example, high-level user commands such as desired vehicle positions are sent to a planner which computes the trajectories for the vehicles which are sent to the Matlab control nodes via IPC. IPC provides flexible message passing and uses TCP/IP sockets to send data between processes.

Each Matlab control node is associated with a radio module containing a 900 MHz and 2.4 GHz Zigbee transceivers, which is used to communicate with all the vehicles in its group. The radio module sends control commands to several vehicles, up to five in this work. Each vehicle operates on a separate channel and the radio module hops between the frequencies for each quadrotor, sending out commands to each vehicle at 100 Hz. The radio modules can also simultaneously receive high bandwidth feedback from the vehicles, making use of the two independent transceivers.

B. Formation Flight

In Fig. 8 we present data for a team of four quadrotors following a trajectory as a formation. The group formation error is significantly larger the the local error. The local x and y errors are always less than 3 cm while the formation x error is as large 11 cm. This data is representative of all formation trajectory following data because all vehicles are nominally the same and are running the same controller with the same gains. Therefore, even though the deviation from the desired trajectory may be large, the relative position error within the group is small.

In Fig. 9(a) we show average error data for 20 vehicles flying in the grid formation shown in Fig. 1. For this experiment the vehicles were controlled to hover at a height of 1.3 meters for at least 30 seconds at several quadrotor-center-to-quadrotor-center grid spacing distances. The air disturbance created from the downwash of all 20 vehicles is significant and causes the tracking performance to be worse for any vehicle in this formation than for an individual vehicle in still air as presented in 4. However, as shown in Fig. 9(a), the separation distance did not have any affect on the hovering performance. Note that at 35 cm grid spacing the nominal distance between propeller tips is about 14 cm.

C. Time-Separated Trajectory Following

In Fig. 9(b) we show a team of 16 vehicles following a cyclic *figure eight* pattern. The time to complete the entire cycle is t_c and the vehicles are equally spaced in time along the trajectory at time increments of $\frac{t_c}{16}$. In order to guaruntee collision-free trajectories at the intersection, vehicles spend $\frac{15}{32}t_c$ in one loop of the trajectory and $\frac{17}{32}t_c$ in the other.

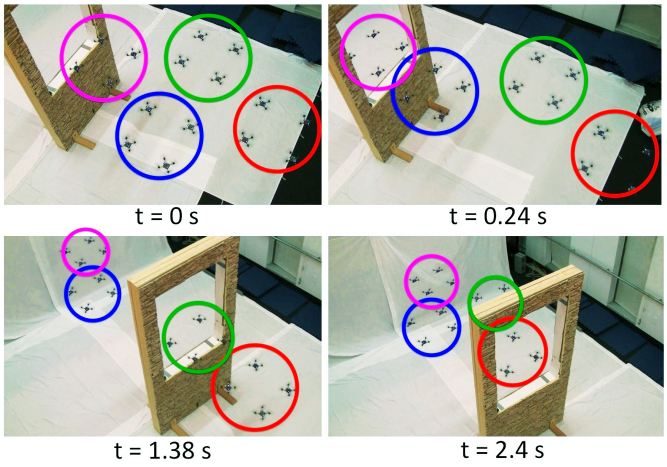


Fig. 10. Four groups of four quadrotors flying through a window

A trajectory that satisfies these timing constraints and has some specified velocity at the intersection point (with zero acceleration and jerk) is generated using the optimization-based method for a single vehicle described in [17].

D. MIQP trajectories

In this paper, we use a branch and bound solver [1] to solve the MIQP trajectory generation problem. The solving time for the MIQP is an exponential function of the number of binary constraints and also the geometric complexity of the environment. The first solution is often delivered within seconds but finding the true optimal solution and a certificate of optimality can take as long as 20 minutes on a 3.4Ghz Core-i7 Quad-Core desktop machine for the examples presented here.

1) Planning for Groups within a Team: In Fig. 10 we show snapshots from an experiment for four groups of four quadrotors transitioning from one side of a gap to the other. Note that in this example the optimal goal assignment is performed at the group-level.

2) Planning for Sub-Regions: In Fig. 11 we show snapshots from an experiments with 16 vehicles transitioning from a planar grid to a three-dimensional helix and pyramid. Directly using the MIQP approach to generate trajectories for 16 vehicles is not practical. Therefore, in both experiments the space is divided into two regions and separate MIQPs with 8 vehicles each are used to generate trajectories for vehicles on the left and right sides of the formation. Note that, in general, the formations do not have to be symmetric but here we exploit the symmetry and only solve a single MIQP for 8 vehicles for these examples. Optimal goal assignment is used so that the vehicles collectively choose their goals to minimize the total cost.

VII. Conclusion

In this paper we describe the design, manufacture, modeling and control of a micro quadrotor that has a 73 gram mass and is 21 cm in diameter, and the architecture and software for coordinating a team of micro quadrotors with experimental

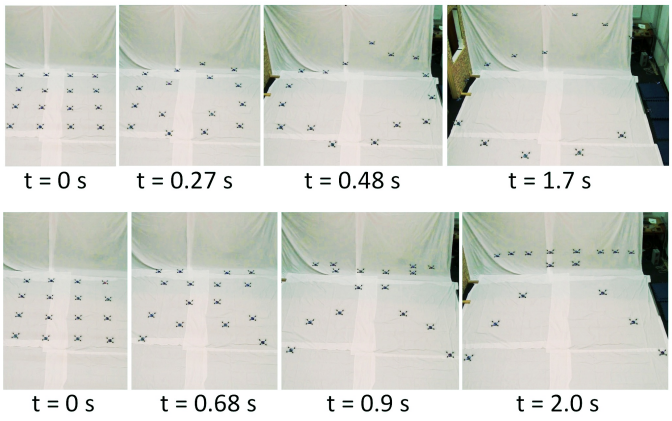


Fig. 11. A team of sixteen vehicles transitioning from a planar grid to a three-dimensional helix (top) and pyramid (bottom)

results. While our quadrotors rely on an external localization system for position estimation and therefore cannot be truly decentralized at this stage, these results represent the first step toward the development of a swarm of micro quadrotors. The small size is shown to facilitate agility and the ability to fly in close proximity with less than one body length separation. Mixed integer quadratic programming techniques are used to coordinate twenty micro quadrotors in known three-dimensional environments with obstacles.

The videos of all experiments are available at http://youtu.be/50Fdi7712KQ

References

[1] IBM ILOG CPLEX V12.1: Users manual for CPLEX, International Business Machines Corporation, 2009.
[2] Ascending Technologies, GmbH. http://www.asctec.de.
[3] Boost C++ Libraries. http://www.boost.org.
[4] Inter Process Communication. http://www.cs.cmu.edu/ ipc/.
[5] Vicon Motion Systems, Inc. http://www.vicon.com.
[6] Aeroenvironment. Aeroenvironment nano hummingbird, August 2011. Online: http://www.avinc.com/nano.
[7] C. Anderson and N. R. Franks. Teams in animal societies. *Behavioral Ecology*, 12(5):534540, 2001.
[8] R. W. Beard, J. Lawton, and F. Y. Hadaegh. A coordination architecture for spacecraft formation control. *IEEE Trans. Control Syst. Technol.*, 9(6):777–790, November 2001.
[9] C. Bermes. *Design and dynamic modeling of autonomous coaxial micro helicopters*. PhD thesis, ETH Zurich, Switzerland, 2010.
[10] S. Bouabdallah. *Design and Control of Quadrotors with Applications to Autonomous Flying*. PhD thesis, Ecole Polytechnique Federale de Lausanne, Lausanne, Switzerland, February 2007.
[11] F. Bullo, J. Cortés, and S. Martínez. *Distributed Control of Robotic Networks: A Mathematical Approach to Motion Coordination Algorithms*. Applied Mathematics Series. Princeton University Press, 2009.

[12] J. P. Desai, J. P. Ostrowski, and V. Kumar. Modeling and control of formations of nonholonomic mobile robots. *IEEE Trans. Robot.*, 17(6):905–908, December 2001.

[13] M. Egerstedt and X. Hu. Formation constrained multi-agent control. *IEEE Trans. Robot. Autom.*, 17(6):947–951, December 2001.

[14] Dario Floreano, Jean-Christophe Zufferey, Adam Klaptocz, Jrg Markus Germann, and Mirko Kovac. Aerial Locomotion in Cluttered Environments. In *Proceedings of the 15th International Symposium on Robotics Research*, 2011.

[15] V. Kumar and N. Michael. Opportunities and challenges with autonomous micro aerial vehicles. In *International Symposium on Robotics Research*, 2011.

[16] 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 Intl. Conf. on Robot. and Autom.*, pages 1642–1648, Anchorage, AK, May 2010.

[17] D. Mellinger and V. Kumar. Minimum snap trajectory generation and control for quadrotors. In *Proc. of the IEEE Intl. Conf. on Robot. and Autom.*, pages 2520–2525, Shanghai, China, May 2011.

[18] D. Mellinger, N. Michael, and V. Kumar. Trajectory generation and control for precise aggressive maneuvers. In *Int. Symposium on Experimental Robotics*, December 2010.

[19] N. Michael and V. Kumar. Control of ensembles of aerial robots. *Proc. of the IEEE*, 99(9):1587–1602, September 2011.

[20] N. Michael, J. Fink, and V. Kumar. Cooperative manipulation and transportation with aerial robots. *Auton. Robots*, 30(1):73–86, January 2011.

[21] J. Parrish and W. Hamner, editors. *Animal Groups in Three Dimensions*. Cambridge University Press, New York, 1997.

[22] D. Pines and F. Bohorquez. Challenges facing future micro air vehicle development. *AIAA J. of Aircraft*, 43 (2):290–305, 2006.

[23] Tom Schouwenaars, Bart DeMoor, Eric Feron, and Jonathan How. Mixed integer programming for multi-vehicle path planning. In *European Control Conference*, pages 2603–2608, 2001.

[24] Tom Schouwenaars, Andrew Stubbs, James Paduano, and Eric Feron. Multi-vehicle path planning for non-line of sight communication. In *American Control Conference*, 2006.

[25] S. Shen, N. Michael, and V. Kumar. Autonomous multi-floor indoor navigation with a computationally constrained MAV. In *Proc. of the IEEE Intl. Conf. on Robot. and Autom.*, pages 20–25, Shanghai, China, May 2011.

[26] H. Tanner, A. Jadbabaie, and G. J. Pappas. Flocking in fixed and switching networks. *IEEE Trans. Autom. Control*, 52(5):863–868, May 2007.

[27] Jur van den Berg, Stephen J. Guy, Ming C. Lin, and Dinesh Manocha. Reciprocal n-body collision avoidance. In *INTERNATIONAL SYMPOSIUM ON ROBOTICS RESEARCH*, 2009.

[28] C. H. Wolowicz, J. S. Bowman, and W. P. Gilbert. Similitude requirements and scaling relationships as applied to model testing. Technical report, NASA, August 1979.