The GRASP Multiple Micro UAV Testbed

Nathan Michael, Daniel Mellinger, Quentin Lindsey, and Vijay Kumar
GRASP Laboratory
University of Pennsylvania
Philadelphia, Pennsylvania 19104-6228
Email: {nmichael, dmel, quentinl, kumar}@grasp.upenn.edu

I. Introduction

In the last five years, advances in materials, electronics, sensors and batteries have fueled a growth in the development of Micro Unmanned Aerial Vehicles (MAVs) that are between 0.1-0.5 meters in length and 0.1-0.5 kilograms in mass [1]. A few groups have built and analyzed MAVs in the 10 cm range [2, 3]. One of the smallest is the Picoflyer with a 60 mm propellor diameter and a mass of 3.3 grams [4]. Platforms in the 50 cm range are more prevalent with several groups having built and flown systems of this size [5]–[7]. In fact, there are several commercially available RC helicopters and research grade helicopters in this size range [8].

Testbeds are especially important for multi-robot research with MAVs as they are often agile and influenced by the dynamics of the environment and coupled to the performance of the other vehicles in complex and hard-to-predict ways. Several groups have outdoor multi UAV testbeds [9, 10] as well as indoor multi-quadrotor environments [11]. The Flying Machine Arena at ETH Zurich is impressive because of its large size ($10 \times 10 \times 10\,\mathrm{m}^3$) with protective netting enclosing the workspace; enabling impressive aerial aerobatics research [12].

Here we address the development of the GRASP Multiple MAV testbed to support research on coordinated, dynamic flight of MAVs with broad applications to reconnaissance, surveillance, manipulation and transport. Our main goal in the paper is to describe our approach to modeling, control and integrating off-the-shelf MAVs with a vignette illustrating implementations of multi-robot control algorithms. We first describe our technical approach to designing the testbed in Sect. II. In Sect. III, we describe the dynamic model used to simulate the system. In Sects. IV–V, we describe controllers implemented on the platform as well as the software enabling experiment design and interaction with the system. We present a multi MAV interaction characterization and experimental results with multiple MAVs performing group formation control in Sect. VI.

II. Technical Approach

Our research focuses on developing new control methods and algorithms for MAVs. We are interested in coordinating actions with multiple MAVs and methods for dealing with the interactions between them.

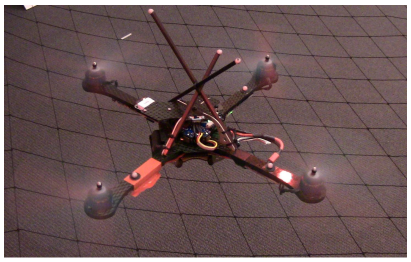


Fig. 1. A quadrotor platform with VICON markers.

A. *Quadrotor Platform*

The design, construction, and testing of a MAV requires a large amount of time. Since this work is not in the scope of our core research goals, we have chosen to purchase off-the-shelf platforms. We chose the Hummingbird quadrotor sold by Ascending Technologies, GmbH [8] shown in Fig. 1 for several reasons. For navigation in highly constrained spaces, helicopters offer the advantage of hovering, which fixed wing aircraft cannot. The Hummingbird fits the physical size requirements: a tip-to-tip wingspan of 55 cm, a height of 8 cm, and a weight of about 500 grams including a battery. The platform is durable enough to survive most crashes while the blades are soft enough not to cause damage during such event. Furthermore, the 20 minute battery life and 200 g payload capacity are also advantageous.

At the software level, the Hummingbird provides additional space on the onboard microcontroller for user code and also offers high-level and low-level command interfaces. The high-level interface allows users to command desired roll, pitch, and yaw angles and the net thrust. The Hummingbird also offers a low-level interface which allows the user to directly set the setpoints for the speeds of the four rotors. This interface allows more precise control over the quadrotor and the development of our own control algorithms.

B. VICON *Motion Capture System*

The VICON Motion Capture System provides a state estimate for the quadrotor, which is nearly ground truth [13]. The VICON system offers three important features. First, *it is fast*: the system can be run at or below 375 Hz. Second, *it is precise*: experimental tests show that the deviations of position estimates for single static markers are on the order of 50 microns which is well beyond the requirements for flight. Lastly, *it is robust*: the VICON *Tracker* software assumes that markers in models are rigidly attached which enables the

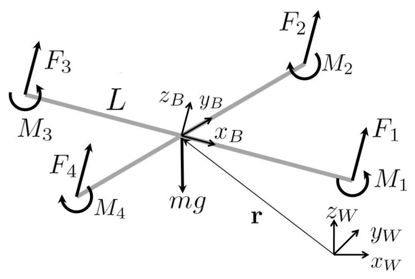


Fig. 2. Coordinate systems and forces/moments acting on the quadrotor frame.

system to maintain tracking even if all but one camera are occluded. Using this software, tracking of quadrotors is rarely lost, even during extreme situations such as fast maneuvers (speeds of 3.5 $\frac{\text{m}}{\text{s}}$, accelerations of 15 $\frac{\text{m}}{\text{s}^2}$ and angular speeds of 1000 $\frac{\circ}{\text{s}}$).

C. Software and Integration

At the core of our software infrastructure is a finite-state machine encoding a hybrid system for constructing experiments with a single or multiple quadrotors (detailed in Sect. V). Each mode of the hybrid system consists of closed loop controllers. The control software is integrated with the motion capture system and accessed via ROS, an open-source, meta-operating system [14] developed and supported by Willow Garage and widely used in the robotics community.

A key consideration in the design of the testbed is the distribution of computation between onboard and external processing and communication between robots and with external systems. The most promising solution we found for onboard processing is the Overo *Fire* board sold by Gumstix [15]. The board is equipped with a 600 MHz ARM processor and runs ROS through cross-compilation. While the board is also equipped with 802.11 and Bluetooth communication options, we found that Zigbee incurs the least latency in communication between robots and external systems. ROS greatly simplifies the transition between onboard and external computation and only requires a few minutes to complete.

D. Simulation

Many of the control algorithms and methods developed in our lab can be designed and tested in simulation before implementation on the physical system. For this reason we developed and characterized a high fidelity dynamic model of the quadrotor. Section III describes the model, and its integration into the control structure is discussed in Sect. V-C.

III. MODELING

A. Dynamic Model

The coordinate systems and free body diagram for the quadrotor are shown in Fig. 2. The world frame, $\mathcal{W}$, is defined by axes x_W, y_W, and z_W with z_W pointing upward. The body frame, $\mathcal{B}$, is attached to the center of mass of the quadrotor with x_B coinciding with the preferred forward direction and z_B perpendicular to the plane of the rotors pointing vertically up during perfect hover (see Fig. 2). Rotor 1 is on the positive x_B-axis, 2 on the positive y_B-axis, 3 on the negative x_B-axis, 4 on the negative y_B-axis. We use $Z-X-Y$ Euler angles to model the rotation of the quadrotor in the world frame. To get from $\mathcal{W}$ to $\mathcal{B}$, we first rotate about z_W by the yaw angle, ψ, then rotate about the intermediate x-axis by the roll angle, ϕ, and finally rotate about the y_B axis by the pitch angle, θ. The rotation matrix for transforming coordinates from $\mathcal{B}$ to $\mathcal{W}$ is given by

$$R = \begin{bmatrix} c\psi c\theta - s\phi s\psi s\theta & -c\phi s\psi & c\psi s\theta + c\theta s\phi s\psi \\ c\theta s\psi + c\psi s\phi s\theta & c\phi c\psi & s\psi s\theta - c\psi c\theta s\phi \\ -c\phi s\theta & s\phi & c\phi c\theta \end{bmatrix},$$

where $c\theta$ and $s\theta$ denote $\cos(\theta)$ and $\sin(\theta)$, respectively, and similarly for ϕ and ψ. 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 forces from each of the rotors, F_i, in the z_B direction. The equations governing the acceleration of the center of mass are

$$m\ddot{\mathbf{r}} = \begin{bmatrix} 0 \\ 0 \\ -mg \end{bmatrix} + R \begin{bmatrix} 0 \\ 0 \\ \Sigma F_i \end{bmatrix}. \tag{1}$$

The components of angular velocity of the robot in the body frame are p, q, and r. These values are related to the derivatives of the roll, pitch, and yaw angles according to

$$\begin{bmatrix} p \\ q \\ r \end{bmatrix} = \begin{bmatrix} c\theta & 0 & -c\phi s\theta \\ 0 & 1 & s\phi \\ s\theta & 0 & c\phi c\theta \end{bmatrix} \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix}.$$

In addition to forces, each rotor produces a moment perpendicular to the plane of rotation of the blade, M_i. Rotors 1 and 3 rotate in the $-z_B$ direction while 2 and 4 rotate in the z_B direction. Since the moment produced on the quadrotor is opposite to the direction of rotation of the blades, M_1 and M_3 act in the z_B direction while M_2 and M_4 act in the $-z_B$ direction. We let L be the distance from the axis of rotation of the rotors to the center of the quadrotor. The moment of inertia matrix referenced to the center of mass along the $x_B - y_B - z_B$ axes, I, is found by weighing individual components of the quadrotor and building a physically accurate model in SolidWorks. The angular acceleration determined by the Euler equations is

$$I \begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} = \begin{bmatrix} L(F_2 - F_4) \\ L(F_3 - F_1) \\ M_1 - M_2 + M_3 - M_4 \end{bmatrix} - \begin{bmatrix} p \\ q \\ r \end{bmatrix} \times I \begin{bmatrix} p \\ q \\ r \end{bmatrix}. \tag{2}$$

B. Motor Model

Each rotor has an angular speed ω_i and produces a vertical force F_i according to

$$F_i = k_F \omega_i^2. \tag{3}$$

Experimentation with a fixed rotor at steady-state shows that $k_F \approx 6.11 \times 10^{-8} \frac{\text{N}}{\text{rpm}^2}$. The rotors also produce a moment according to

$$M_i = k_M \omega_i^2. \tag{4}$$

The constant, k_M, is determined to be about $1.5 \times 10^{-9} \frac{\text{Nm}}{\text{rpm}^2}$ by matching the performance of the simulation to the real system.

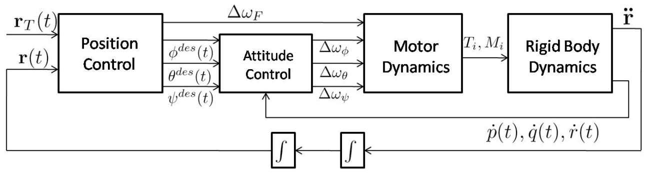


Fig. 3. The nested control loops for position and attitude control.

The results of a system identification exercise suggest that the rotor speed is related to the commanded speed by a first-order differential equation

$$\dot{\omega_i} = k_m(w_i^{\text{des}} - w_i).$$

This motor gain, k_m, is found to be about $20\,\text{s}^{-1}$ by matching the performance of the simulation to the real system. The desired angular velocities, ω_i^{des}, are limited to a minimum and maximum value determined through experimentation to be approximately $1200\,\text{rpm}$ and $7800\,\text{rpm}$.

IV. Robot Controllers

Each robot is controlled independently by nested feedback loops as shown in Fig. 3. The inner attitude control loop uses onboard accelerometers and gyros to control the roll, pitch, and yaw and runs at approximately $1\,\text{kHz}$ [5], while the outer position control loop uses estimates of position and velocity of the center of mass to control the trajectory in three dimensions. Similar nesting of control loops is presented in previous works [5]–[7, 16, 17].

Our controllers are derived by linearizing the equations of motion and motor models (1 – 4) at an operating point that corresponds to the nominal hover state, $\mathbf{r} = \mathbf{r}_0$, $\theta = \phi = 0$, $\psi = \psi_0$, $\dot{\mathbf{r}} = 0$, and $\dot{\phi} = \dot{\theta} = \dot{\psi} = 0$, where the roll and pitch angles are small ($c\phi \approx 1$, $c\theta \approx 1$, $s\phi \approx \phi$, and $s\theta \approx \theta$). At this hover state, the nominal thrusts from the propellers must satisfy

$$F_{i,0} = \frac{mg}{4},$$

and the motor speeds are given by

$$\omega_{i,0} = \omega_h = \sqrt{\frac{mg}{4k_F}}.$$

A. Attitude Control

We now present an attitude controller to track trajectories in $SO(3)$ that are close to the nominal hover state where the roll and pitch angles are small. From (2), substituting in the relationships between angular velocities of the rotors and forces and moments, (3) and (4),

$$I_{xx}\dot{p} = Lk_F(\omega_2^2 - \omega_4^2) - qr(I_{zz} - I_{yy}) \tag{5a}$$

$$I_{yy}\dot{q} = Lk_F(\omega_3^2 - \omega_1^2) - pr(I_{xx} - I_{zz}) \tag{5b}$$

$$I_{zz}\dot{r} = k_M(\omega_1^2 - \omega_2^2 + \omega_3^2 - \omega_4^2). \tag{5c}$$

Note that the products of inertia are small (ideally, they are zero because the axes are close to the principal axes) and $I_{xx} \approx I_{yy}$ because of the symmetry. We assume the component of the angular velocity in the z_B direction, r, is small so the rightmost terms in (5a) and (5b) which are products involving r are small compared to the other terms. The vector of desired rotor speeds can be written as a linear combination of four terms

$$\begin{bmatrix}\omega_1^{\text{des}}\\ \omega_2^{\text{des}}\\ \omega_3^{\text{des}}\\ \omega_4^{\text{des}}\end{bmatrix} = \begin{bmatrix}1 & 0 & -1 & 1\\ 1 & 1 & 0 & -1\\ 1 & 0 & 1 & 1\\ 1 & -1 & 0 & -1\end{bmatrix}\begin{bmatrix}\omega_h + \Delta\omega_F\\ \Delta\omega_\phi\\ \Delta\omega_\theta\\ \Delta\omega_\psi\end{bmatrix}, \tag{6}$$

where the nominal rotor speed required to hover in steady state is ω_h, and the deviations from this nominal vector are $\Delta\omega_F$, $\Delta\omega_\phi$, $\Delta\omega_\theta$, and $\Delta\omega_\psi$. $\Delta\omega_F$ results in a net force along the z_B axis, while $\Delta\omega_\phi$, $\Delta\omega_\theta$, and $\Delta\omega_\psi$ produce moments causing roll, pitch, and yaw, respectively. This is similar to the approach described in [5].

Now we linearize (5a), (5b), and (5c) about the hovering operating point and write the desired angular accelerations in terms of the new control inputs

$$\dot{p}^{\text{des}} = \frac{4k_F L\omega_h}{I_{xx}}\Delta\omega_\phi,$$
$$\dot{q}^{\text{des}} = \frac{4k_F L\omega_h}{I_{yy}}\Delta\omega_\theta,$$
$$\dot{r}^{\text{des}} = \frac{8k_M\omega_h}{I_{zz}}\Delta\omega_\psi.$$

As near the nominal hover state $\dot{\phi} \approx p$, $\dot{\theta} \approx q$, and $\dot{\psi} \approx r$, we use proportional derivative control laws that take the form

$$\begin{aligned}\Delta\omega_\phi &= k_{p,\phi}(\phi^{\text{des}} - \phi) + k_{d,\phi}(p^{\text{des}} - p)\\ \Delta\omega_\theta &= k_{p,\theta}(\theta^{\text{des}} - \theta) + k_{d,\theta}(q^{\text{des}} - q)\\ \Delta\omega_\psi &= k_{p,\psi}(\psi^{\text{des}} - \psi) + k_{d,\psi}(r^{\text{des}} - r).\end{aligned} \tag{7}$$

Substituting (7) into (6) yields the desired rotor speeds.

B. Position Control

Here we present two representative position control methods that use the roll and pitch angles as inputs via a method similar to a backstepping approach [18]. The first, a hover controller, is used for station-keeping or maintaining the position at a desired x, y, and z location. The second tracks a trajectory in three dimensions.

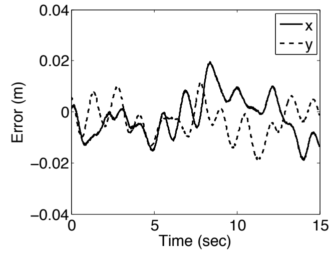
1) Hover Controller: Here we use pitch and roll angle to control position in the x_W and y_W plane, $\Delta\omega_\psi$ to control yaw angle, and $\Delta\omega_F$ to control position along z_W. We let $\mathbf{r}_T(t)$ and $\psi_T(t)$ be the trajectory and yaw angle we are trying to track. Note that $\psi_T(t) = \psi_0$ for the hover controller. The command accelerations, $\ddot{r}_i^{\text{des}}$, are calculated from PID feedback of the position error, $e_i = (r_{i,T} - r_i)$, as

$$\begin{aligned}(\ddot{r}_{i,T} - \ddot{r}_i^{\text{des}}) \quad &+ \quad k_{d,i}(\dot{r}_{i,T} - \dot{r}_i) + k_{p,i}(r_{i,T} - r_i)\\ &+ \quad k_{i,i}\int(r_{i,T} - r_i) = 0,\end{aligned}$$

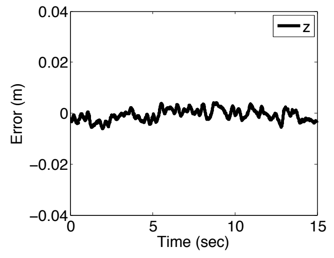
where $\dot{r}_{i,T} = \ddot{r}_{i,T} = 0$ for hover.

Then we linearize (1) to get the relationship between the desired accelerations and roll and pitch angles

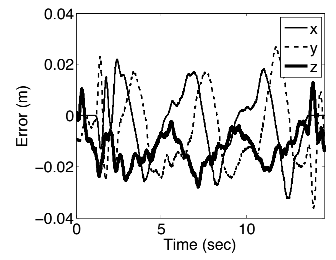
$$\ddot{r}_1^{\text{des}} = g(\theta^{\text{des}}\cos\psi_T + \phi^{\text{des}}\sin\psi_T)$$
$$\ddot{r}_2^{\text{des}} = g(\theta^{\text{des}}\sin\psi_T - \phi^{\text{des}}\cos\psi_T)$$
$$\ddot{r}_3^{\text{des}} = \frac{8k_F\omega_h}{m}\Delta\omega_F.$$



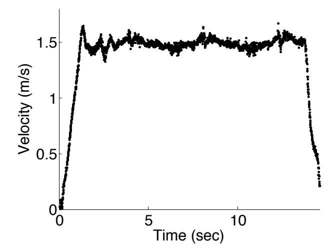
(a) Errors in the x and y directions.



(b) Error in the z direction.



(c) Errors in position.



(d) Velocity history.

Fig. 4. Representative results with the hover controller (Figs. 4(a)–4(b)). Experimental results from the 3D trajectory controller tracking a circle of radius 1 m with the axis making a 45° angle from the vertical at 1.5 $\frac{m}{s}$ (Figs. 4(c)–4(d)).

These relationships are inverted to compute the desired roll and pitch angles for the attitude controller, from the desired accelerations, as well as $\Delta\omega_F$

$$\phi^{\text{des}} = \frac{1}{g}(\ddot{r}_1^{\text{des}} \sin\psi_T - \ddot{r}_2^{\text{des}} \cos\psi_T) \tag{9a}$$

$$\theta^{\text{des}} = \frac{1}{g}(\ddot{r}_1^{\text{des}} \cos\psi_T + \ddot{r}_2^{\text{des}} \sin\psi_T) \tag{9b}$$

$$\Delta\omega_F = \frac{m}{8k_F\omega_h}\ddot{r}_3^{\text{des}}. \tag{9c}$$

The position control loop for the hover controller runs at 100 Hz, while the inner attitude control loop runs at 1 kHz. There is the usual trade-off in optimizing the control gains between speed of response and stability. Experimental results show (see the representative trial in Figs. 4(a)–4(b)) for a tightly optimized "stiff" controller the horizontal positioning errors are within 2 cm and the error in the vertical direction is always less than 0.6 cm. However, this set of gains leads to a relatively small basin of attraction. By optimizing the gains for a softer response we can increase the size of this basin of attraction. We can experimentally characterize this basin by perturbing the quadrotor from the hover state, and measuring the response of the hover controller. We found the robot to be quite robust if we used the "softer" controller, allowing it to recover from disturbances as large as 1.5 m (3 body lengths) in the horizontal direction and 2.0 m (4 body lengths) in the vertical direction, pitch or roll angle errors of 60 °, and velocity errors of up to 3.0 $\frac{\text{m}}{\text{s}}$.

2) 3D Trajectory Control: The 3D Trajectory Controller is used to follow three-dimensional trajectories with modest accelerations so the near-hover assumptions hold. We use an approach similar to that described in [7] but extend it from 2D to 3D trajectories. We have a method for calculating the closest point on the trajectory, $\mathbf{r}_T$, to the the current position, $\mathbf{r}$. Let the unit tangent vector of the trajectory associated with that point be $\hat{\mathbf{t}}$ and the desired velocity vector be $\dot{\mathbf{r}}_T$. We define the position and velocity errors as

$$\mathbf{e}_p = ((\mathbf{r}_T - \mathbf{r}) \cdot \hat{\mathbf{n}})\hat{\mathbf{n}} + ((\mathbf{r}_T - \mathbf{r}) \cdot \hat{\mathbf{b}})\hat{\mathbf{b}}$$

and

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

Note that here we ignore position error in the tangent direction by only considering position error in the normal, $\hat{\mathbf{n}}$, and binormal, $\hat{\mathbf{b}}$, directions.

We calculate the commanded acceleration, $\ddot{r}_i^{\text{des}}$, from PD feedback of the position and velocity errors:

$$\ddot{r}_i^{\text{des}} = k_{p,i}e_{i,p} + k_{d,i}e_{i,v} + \ddot{r}_{i,T}.$$

Note that the $\ddot{r}_{i,T}$ terms represent feedforward terms on the desired accelerations. At low accelerations these terms can be ignored but at larger accelerations they can significantly improve controller performance. Finally we use (9a), (9b), and (9c) to compute the desired roll and pitch angles as well as $\Delta\omega_F$.

Here we present results for the 3D trajectory tracking controller. The desired trajectory is a circle with a 1 m radius tilted at 45° from horizontal. The quadrotor is controlled to start from rest and accelerate at a rate of 1.5 $\frac{\text{m}}{\text{s}^2}$ to the lowest point on the circle, complete three revolutions around the circle at 1.5 $\frac{\text{m}}{\text{s}}$, and then decelerate to a stop at a rate of 1.5 $\frac{\text{m}}{\text{s}^2}$. The error is shown is Fig. 4(c), note that the x and y errors are always less than 4 cm and the z error is always less than 3 cm. The speed for this trajectory is shown in Fig. 4(d).

V. Control Software and Integration

Working with many MAVs at one time is challenging due to the dynamic capabilities of the robot, the inability to predict complex interactions between robots such as aerodynamic effects, and the likelihood of collisions with multiple aerial vehicles flying in close proximity. To this end we focused on the construction of a system that is: *accessible* to general users with limited or significant experimental and programming experience; *easy to use* both in experiment construction and implementation; *transparent* in operation to facilitate evaluation of performance; and sufficiently *flexible* to support most experimental requirements. We found that these needs are met through the construction of experiments defined by a finite-state machine encoding a hybrid system with interfaces accessible via C++, Python, and Matlab.

A. Finite-state Machine

Experiment design for evaluating control algorithms with multiple MAVs generally have similar descriptions. Nearly all experiments require the robots start in the workspace in a stable configuration. After the robots are initialized, any control law may be tested, but during experiments there must always be a method to "pause" an experiment or handle critical failures. Further, after the completion of the experiment the robots must be removed from the workspace in a predictable and safe manner. Additionally, the demands of different experiments may vary. For example, as a representative experiment, we may wish to test a trajectory tracking control law similar to that in Sect. IV-B.2 but with multiple robots. Toward this experiment, the user must develop a process for deploying the robots, engaging the robots to start the controller, pausing or restarting the controller as required, and landing the robots. Ideally, a single person must be able to conduct and oversee all aspects of the experiment.

We approach the design of such an experiment via the construction of a sequence of states for each robot. Each state is a *behavior* or mode that defines a particular segment of the experiment ranging from turning the robot motors on or off to dynamic flight along a trajectory. Each behavior has associated entry and exit actions as well as transitional checks (behaviors have access to the definition of the prior and next behavior consistent with the semantics of a hybrid system). These behaviors are sequenced into a program that is defined *a priori* or at runtime. The program is provided to a finite-state machine (FSM) that drives the feedback controller and monitors the performance of each robot. The FSM may run either on the robot itself or on an external computer and accepts and applies new sequences at any time. Additionally, the transitioning of the FSM may be manually requested at any time.

As an example, consider the representative experiment described above. To perform such an experiment we construct a sequence of behaviors as follows:

1) turn the robot motors on when carried into the workspace;
2) fly to location $\mathbf{r}_1$ with a fixed velocity;
3) hover at $\mathbf{r}_1$ for a fixed duration;
4) run the controller;
5) land at location $\mathbf{r}_2$ with collision avoidance;
6) turn off the motors.

The user sends the sequence to each robot and now only needs to carry the robots into the workspace; the finite-state machine manages the remainder of the experiment. At any time we may wish to pause an experiment so we request that the robot hover at the location. If we wish to resume or restart the experiment we send a new sequence with the appropriate definitions. At any time we can request that the robot land at a particular location by requesting the appropriate sequence of behaviors.

This approach is simple to understand and execute allowing users with differing skill sets to run experiments. Further, over time we continually develop more behaviors that allow for increasing complexity in our experiments. For example, to construct an experiment that evaluates a feedback control law that assumes a first-order kinematic abstraction with velocity control inputs (e.g. $u_i = \dot{r}_i$) requires only a matter of minutes to specify and evaluate. The approach is sufficiently flexible that we are able to test anything from simple hovering control to acrobatics. Clearly, the focus is placed on the behavior and algorithm design, not on the experiment construction and implementation. Working with multiple robots is as easy as running another finite-state machine with the same or different sequence of behaviors that now consider collision avoidance and other relevant multi-robot details.

The integration of the FSM into the infrastructure and ROS allows behaviors to use feedback information from the VICON tracking system and sensors on or off the robot. Additionally, ROS provides C++ and Python implementations, which we augment with our own MATLAB interface (in lieu of the ROS OCTAVE implementation). Via these languages, an individual has the ability to construct experiments using already generated behaviors or by designing his own. At an implementation level, behaviors are simply inherited class objects that must implement required methods. These class objects may be interfaced and adapted to accomodate different programming languages. The only other requirement of a behavior implementation is that if it generates commands to be sent to the robot, these commands must be parameterized by desired attitude, $\{\phi^{\text{des}},\, \theta^{\text{des}},\, \psi^{\text{des}}\}$, desired angular velocities $\{p^{\text{des}},\, q^{\text{des}},\, r^{\text{des}}\}$, and desired thrust, $u_F^{\text{des}} = \omega_h + \Delta\omega_F$.

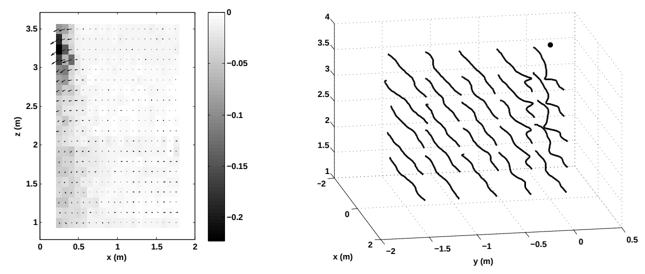
B. Onboard Controller

For experiments that require near hover performance it is possible to control the robot based on only input of the desired attitude and thrust commands. However, dynamic flight and aggressive maneuvers requires a greater control. Additionally, depending on the interaction of the robots, different PD gains result in different levels of performance (e.g. stiffer controllers perform poorly with significant external disturbance). To address all cases we decompose the control problem into two separate control loops: the faster control loop handles attitude control (6, 7) and runs on the robot. This control loop runs on a microprocessor on the robot at 1 kHz with access to IMU data at 300 Hz. The control loop is defined by the desired inputs $\{\phi^{\text{des}},\, \theta^{\text{des}},\, \psi^{\text{des}}\}$, $\{p^{\text{des}},\, q^{\text{des}},\, r^{\text{des}}\}$, u_F^{des} and the associated controller gains $\{k_{p,\phi},\, k_{p,\theta},\, k_{p,\psi}\}$ and $\{k_{d,\phi},\, k_{d,\theta},\, k_{d,\psi}\}$. The slower control loop runs on a system either external to the robot communicating via Zigbee or mounted onboard the robot communicating via serial communications. In either case, the external system generates lower frequency inputs sent to the robot (typically 100 Hz).

C. Quadrotor Simulator

The final key component of our software architecture is the quadrotor simulator. The importance of the quadrotor simulator is two-fold. First, the simulation is highly accurate, which allows us to test algorithms safely and efficiently. Secondly, we can easily switch between the quadrotor simulator and the actual robot. The command messages to which the simulator subscribes are identical to the messages sent to the onboard controller. This feature allows for quick transitions from simulations to experiments.

The quadrotor simulator provides state estimates and emulates the performance and time delays that appear in the actual system. The core of quadrotor simulator is the numerical integration of the 16 state quadrotor model detailed in Sect. III. A limitation is that we do not simulate aerodynamic effects between robots discussed in Sect. VI-A and so we characterize these effects experimentally and assume conservative approximations in our algorithms based on these experiments.



(a) (b)

Fig. 5. Displacement error in the $x_W - z_W$ plane for a quadrotor hovering below a quadrotor hovering at $(-0.03, -0.03, 4.0)$ m (Fig. 5(a)). The trajectories for Quadrotor $\mathbf{B}$ moving parallel to the x_W axis at a desired speed of 1.0 m/s in the proximity of Quadrotor $\mathbf{A}$ hovering at $(-0.03,\ -0.03,\ 4)$ m (location marked by the black dot, Fig. 5(b)).

VI. Group Behaviors

A. Aerodynamic Interactions

To further cooperation among robots, it is essential to have the robots operate in close proximity and maintain constraints on separation. Thus formation flight is a key component technology. However, for three-dimensional flight, the interaction between MAVs goes beyond constraints on relative positions due to specifications on the task and collision avoidance. One quadrotor can influence the dynamic behavior of neighboring quadrotors because of the downwash from its rotors. This effect has consequences in many applications including formation control because additional requirements must be imposed on the relative poses of quadrotors. In order to design algorithms for near-hover controllers that consider these effects, we first need to quantify the extent of the influence. We designed two experiments to measure this disturbance. The purpose of the first experiment is to characterize the aerodynamic effect of a quadrotor hovering above another quadrotor. In the experimental results shown in Fig. 5(a), the first quadrotor, $\mathbf{A}$, uses a hover controller to maintain the position $(-0.03, -0.03, 4.0)$ m. A second quadrotor, $\mathbf{B}$, uses a hover control to execute a sequence of waypoints on a 20×20 grid on the $x_W - z_W$ plane, dwelling 5 seconds at each waypoint. At each waypoint, position data is recorded for a 5 second interval after $\mathbf{B}$'s speed falls below the threshold of $0.005m/s$. In Fig. 5(a), we show the z displacement error in gray scale with the steady-state xz displacement error vector. It is apparent that $\mathbf{A}$'s influence is mostly concentrated to a cylindrical region with a radius of approximately 0.5 m extending to a height of $1.5m$ below the quadrotor. This cylindrical region bounds the volume where the z displacement error is greater than 5 cm. In this cylindrical region, the x displacement error for $\mathbf{B}$ ranges from -0.12 to -0.37 m and z errors from -0.05 to -0.22 m. Although the severe effects are contained in this region, it is clear from the data that there is a larger region in which $\mathbf{B}$ is affected to a lesser extent.

The second experiment characterizes the errors in trajectories of $\mathbf{B}$ induced by $\mathbf{A}$. With the same position for $\mathbf{A}$ as before, $\mathbf{B}$ executes the trajectory controller with a desired speed of 1.0 m/s along a trajectory parallel to the x axis from $x = -1.5$ m to $x = 1.5$ m with the y and z coordinates being chosen from a 5×5 grid on the $y_W - z_W$ plane as shown in Fig. 5(b). Once again, we see a trend similar to the one in Fig. 5(a). Trajectories, which are closer to $\mathbf{A}$, show the largest deviations from the straight line trajectories. We observe that the trajectories close to $\mathbf{A}$ are pushed down and pulled into the downwash of $\mathbf{A}$.

B. Control of Aerial Robot Ensembles

In this section we present experimental results with a team of MAVs controlling to a desired ensemble pose and shape. We begin by briefly detailing the theory behind the control law. We provide the description of the experiment design and review the results. The underlying theory of this section is provided in [19].

1) Statistical Representation and Control of Aerial Robot Formations: We consider a team of N point robots in three-dimensional space with the position of the i^{th} robot denoted as $q_i \in \mathbb{R}^3$. We wish to control the ensemble pose and shape, which is decomposed into a shape space, S, and a Lie group, G, which in our case is $SE(3)$. Define an abstract space, M, whose dimension is smaller and independent of the number of robots, by a smooth, differentiable map

$$\phi : Q \to M, \quad \phi(q) = \boldsymbol{x}, \tag{10}$$

where ϕ is a mapping of the higher-dimensional state space of the team of robots $q \in Q$ to the lower-dimensional abstract state $\boldsymbol{x} \in M$.

The abstract description of the team of robots, $\boldsymbol{x}$, is given by the position and orientation of the team, g, and the shape s. Therefore, ϕ defines a transformation from the space of $Q \in \mathbb{R}^{3N}$ to the lower and fixed dimensional space $\boldsymbol{x} \in M$ where $M \in \mathbb{R}^9$ is defined by the six-dimensional pose and three-dimensional shape of the formation.

We define the mean of the group by

$$\mu = \frac{1}{N}\sum_{i=1}^{N} q_i, \tag{11}$$

and the orientation to be such that the coordinates of the robots in the local frame, $p_i = [x_i,\ y_i,\ z_i]^{\mathrm{T}}$, satisfy

$$\sum_{i=1}^{N} x_i y_i = \sum_{i=1}^{N} x_i z_i = \sum_{i=1}^{N} y_i z_i = 0.$$

Here the position of each robot is $p_i = R^{\mathrm{T}}(q_i - \mu)$ with respect to the frame fixed to the group of robots. The shape space, $s = (s_1,\ s_2,\ s_3)$, is defined by

$$s_1 = \kappa\mathcal{I}_{11}, \quad s_2 = \kappa\mathcal{I}_{22}, \quad s_2 = \kappa\mathcal{I}_{33}, \tag{12}$$

where $\kappa > 0$ and

$$\mathcal{I} = \sum_{i=1}^{N} p_i\, p_i^T = \begin{bmatrix} \mathcal{I}_{11} & 0 & 0 \\ 0 & \mathcal{I}_{22} & 0 \\ 0 & 0 & \mathcal{I}_{33} \end{bmatrix}. \tag{13}$$

Choosing $\kappa = \frac{1}{N-1}$ gives the shape variables a geometric interpretation as the semi-major and semi-minor axes for a concentration ellipse for a group of robots whose coordinates are chosen to satisfy a normal distribution.

Using the natural kinetic energy metric on Q, it is possible to derive the optimal velocity (tangent vector) at any point $q \in Q$ for a desired $\dot{\boldsymbol{x}}$ at the corresponding point $\boldsymbol{x} = \phi(q) \in M$. It was shown in [19] that this input, $u^\star$, for the system can be found by considering the time derivative of the transformation described by (10),

$$d\phi \dot{q} = \dot{\boldsymbol{x}}. \tag{14}$$

Thus, the minimum-energy solution satisfying (14) is obtained using the Moore-Penrose Inverse:

$$u^\star = d\phi^{\mathrm{T}}(d\phi d\phi^{\mathrm{T}})^{-1}\dot{\boldsymbol{x}}. \tag{15}$$

In [19], the individual control law for each agent is found to be

$$\begin{aligned} u_i^\star = \dot{q}_i = \dot{\mu} &+ \frac{s_2 - s_3}{s_2 + s_3} T_3^2 (q_i - \mu)\omega_1 + \frac{s_3 - s_1}{s_1 + s_3} T_3^1 (q_i - \mu)\omega_2 \\ &+ \frac{s_1 - s_2}{s_1 + s_2} T_2^1 (q_i - \mu)\omega_3 + \sum_{k=1}^{3} \frac{\dot{s}_k}{2 s_k} H_k (q_i - \mu), \end{aligned} \tag{16}$$

with $(\omega_1, \omega_2, \omega_3)$ as the angular velocity change of the ensemble shape and

$$H_j^i = R e_i e_j{}^{\mathrm{T}} R^{\mathrm{T}}, \quad T_j^i = H_j^i + H_i^j,$$

where $[e_1\ e_2\ e_3] = I_3$ and $i,\, j = \{1,\, 2,\, 3\}$.

Note that the controller for each agent is only dependent on its state q_i and the abstract state $\boldsymbol{x}$. A global observer that is able to acquire knowledge of the state and provide the values of the abstract state is sufficient to control the entire system. We will use the VICON system in the experiments as the global observer.

2) Introducing Inter-Robot Collision Avoidance and Aerodynamic Interaction Effects: From Sect. VI-A, we see that the inter-robot aerodynamic effects are considerable as the robots get closer together but may be avoided by ensuring that the robots maintain at least a distance of separation of $0.5\,\mathrm{m}$ in the x and y directions and $1.5\,\mathrm{m}$ in the z direction. Therefore, we approximate these regions as cylinders with a radius of $0.5\,\mathrm{m}$ and height of $1.5\,\mathrm{m}$ centered about q_i. Assuming the robots start outside the cylindrical region of other robots, we ensure that robots avoid collisions and reduce aerodynamic interactions by requiring that

$$(q_i - q_j) \cdot (\dot{q}_i - \dot{q}_j) \geq 0, \tag{17}$$

for all robots, q_j, that lie on the boundary of the cylinder surrounding q_i.

3) Monotonic Convergence with Interactions: In the absence of collisions and aerodynamic interactions, the easiest way to guarantee convergence to an abstract state $\boldsymbol{x}^{\mathrm{des}}$ is to require the error, $\tilde{\boldsymbol{x}}$, to converge exponentially to zero:

$$\dot{\boldsymbol{x}} = K\, \tilde{\boldsymbol{x}},$$

where K is any positive-definite matrix.

We relax the requirement of exponential convergence to an abstract state and instead of insisting on the minimum-energy solution, (16), find the solution closest to the minimum-energy solution satisfying (17). Additionally, we require that the error in the abstract state decrease *monotonically*

$$\tilde{\boldsymbol{x}}^{\mathrm{T}} K \dot{\boldsymbol{x}} \geq 0. \tag{18}$$

It is shown in [19] that a sufficient condition to satisfy this *monotonic convergence condition* is to require that each robot select inputs that satisfy

$$\tilde{\boldsymbol{x}}^{\mathrm{T}} K d\phi_i \dot{q}_i \geq 0, \tag{19}$$

where $d\phi_i$ is the i^{th} column of the linear mapping in (14). The decentralized control law that selects a control input as close as possible to the minimum-energy controls while satisfying the monotonic convergence inequality and the satisfying the interaction conditions is formulated as a quadratic program:

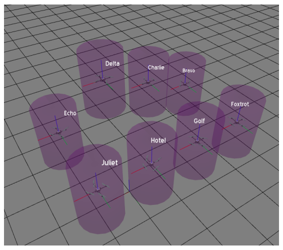
$$u_i = \underset{\hat{u}_i \in U}{\arg\min} \|u_i^\star - \hat{u}_i\|^2, \quad s.t. \quad (17, 18). \tag{20}$$

The convergence and stability properties of (20) as well as the uniqueness and existence of solutions to the program are detailed in [19].

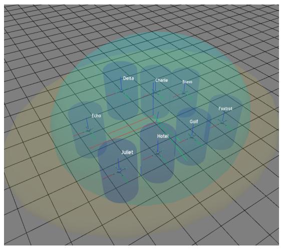
4) Simulation and Experimental Results: We evaluated the ensemble pose and shape control law defined by (20) on a team of eight quadrotor in simulation and four quadrotors in experimentation. For this study, we must consider a control law that requires state feedback of neighboring robots' positions and the current abstract description $\boldsymbol{x}$. Further, we wish to perform waypoint control to desired ensemble states $\boldsymbol{x}_i^{\mathrm{des}}$. The progression of the simulation and experiment follows:

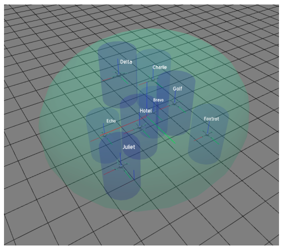
- deploy the robots to an initial configuration (Figs. 6(a) and 6(e));
- hover until ready to evaluate (20) (Fig. 6(b));
- enable (20) on all robots;
- broadcast $\boldsymbol{x}_i^{\mathrm{des}}$ to all robots (Figs. 6(c)–6(d) and Figs. 6(f)–6(h));
- hover in place when evaluation concludes;
- land.

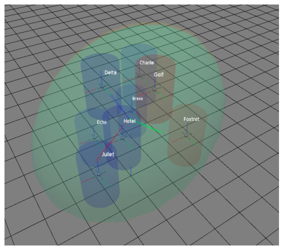
As noted in Sect. V, the complexity of this experiment is greatly reduced by using behaviors from prior work. In fact, the only items specific to this study is the evaluation of (20) on the robots and the broadcasting of $\boldsymbol{x}_i^{\mathrm{des}}$ to the ensemble. Each robot computes u_i based on $\boldsymbol{x}$, $\boldsymbol{x}_i^{\mathrm{des}}$, and the positions of the other robots. As noted above, we are able to account for collision avoidance and aerodynamic effects by an appropriate separation between robots. Figure 5(a) shows that this spacing does not completely eliminate all aerodynamic effects but reduces the impact considerably. With each robot computing their individual controls, u_i, according to (20), we must now apply these inputs to the hardware. We leverage the hover controller described in Sect. IV-B.1 by defining set-points generated by each update of u_i at a fixed interval. We compute u_i at a rate of $10\,\mathrm{Hz}$, transform these inputs to vehicle control inputs at a rate of $100\,\mathrm{Hz}$ via the methods of Sect. IV-B.1, and apply the resulting dynamic commands on the robot's embedded processor at $1\,\mathrm{kHz}$.

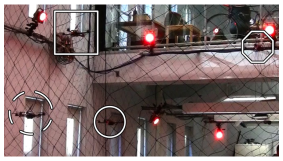


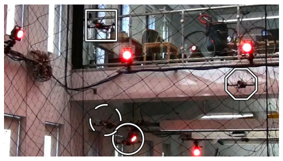
(a)

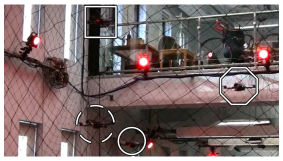

(b)


(c)


(d)


(e)


(f)


(g)

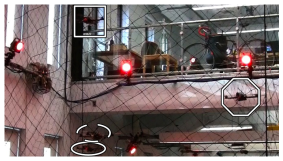

(h)

Fig. 6. An ensemble of eight quadrotors in simulation (Figs. 6(a)–6(d)) and four quadrotors in experimentation (Figs. 6(e)–6(h)) control to $\boldsymbol{x}_i^{\text{des}}$. The aqua and orange ellipsoids represent the desired and current abstract states, respectively, as concentration ellipsoids and the cylinders depict the regions of aerodynamic effects (shown in red if the robots are avoiding other regions and blue otherwise). Unique shapes mark each robot in the experiment images to help identify their positions and correlate motions between images. The full video of the experiment is available in the multimedia attachment.

VII. Conclusion, Discussion, and Future Work

In this work we present the details of our MAV testbed developed to support experimental evaluation of multi-robot aerial control algorithms. A key point of our approach is a finite-state machine encoding a hybrid system that facilitates modular code development and rapid integration with simulated and real hardware. By leveraging tools that automate the experimental process, we lower the required knowledge to use the system and open experimentation up to users with various degrees of experience. Through the use of off-the-shelf platforms and technologies we reduce the time required to acquire and integrate new platforms. Finally, the system design is scalable to a large number of robots determined only by space restrictions and experimental needs.

There are pragmatic limitations to our approach such as our reliance on the Vicon tracking system and limited flight times due to battery performance. Additionally, we only propose the use of a single type of aerial robot platform; the presentation would most certainly have differed with the selection of a fixed-wing vehicle. A final constraint is our limited experimental environment size with a usable volume of approximately $5 \times 4 \times 3.5\,\text{m}^3$.

Our current research interests include planning and control for precise aggressive maneuvers with quadrotors that enable applications such as flying through vertical and horizontal openings and perching. We are also extending the study discussed in Sect. VI-A toward adaptive feedback control laws that consider inter-robot aerodynamic effects and external wind disturbances. While we emphasize the use of the external Vicon system for robot localization, we are currently working on using onboard sensors on a larger quadrotor platform for generating maps and localizing in indoor environments.

References

[1] D. Pines and F. Bohorquez, "Challenges facing future micro air vehicle development," *AIAA Journal of Aircraft*, vol. 43, no. 2, pp. 290–305, 2006.

[2] I. Kroo, F. Prinz, M. Shantz, P. Kunz, G. Fay, S. Cheng, T. Fabian, and C. Partridge, "The mesicopter: A miniature rotorcraft concept, phase ii interim report," 2000.

[3] B. Hein and I. Chopra, "Hover performance of a micro air vehicle: Rotor at low reynolds number," *Journal of the American Helicopter Society*, vol. 52, no. 3, pp. 254–262, July 2007.

[4] "Proxflyer," http://www.proxflyer.com.

[5] 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.

[6] S. Bouabdallah, "Design and control of quadrotors with application to autonomous flying," Ph.D. dissertation, Ecole Polytechnique Federale de Lausanne, 2007.

[7] G. Hoffmann, S. Waslander, and C. Tomlin, "Quadrotor helicopter trajectory tracking control," in *AIAA Guidance, Navigation and Control Conference and Exhibit*, Honolulu, Hawaii, Apr. 2008.

[8] "Ascending technologies," http://www.asctec.de.

[9] A. Richards, J. Bellingham, M. Tillerson, and J. How, "Co-ordination and control of multiple uavs," in *AIAA Guidance, Navigation and Control Conference and Exhibit*, Monterey, CA, Aug. 2002.

[10] D. Shim, H. Kim, and S. Sastry, "A flight control system for aerial robots: Algorithms and experiments," *IFAC Control Engineering Practice*, 2003.

[11] M. Valenti, B. Bethke, G. Fiore, and J. How, "Indoor multi-vehicle flight testbed for fault detection, isolation, and recovery," in *AIAA Guidance, Navigation and Control Conference and Exhibit*, Keystone, CO, Aug. 2006.

[12] S. Lupashin, A. Schllig, M. Sherback, and R. D'Andrea, "A simple learning strategy for high-speed quadrocopter mult-flips," in *Proc. of the IEEE Int. Conf. on Robotics and Automation*, Anchorage, AK, 2010.

[13] "Vicon MX Systems," http://www.vicon.com/products/viconmx.html.

[14] M. Quigley, B. Gerkey, K. Conley, J. Faust, T. Foote, J. Leibs, E. Berger, R. Wheeler, and A. Ng, "ROS: an open-source robot operating system," in *Open-source software workshop of the Int. Conf. on Robotics and Automation*, Kobe, Japan, 2009.

[15] "Gumstix," http://www.gumstix.com.

[16] E. Altug, J. Ostrowski, and C. Taylor, "Control of quadrotor helicopter using dual camera visual feedback," *The Int. Journal of Robotics Research*, vol. 24, no. 5, pp. 329–341, May 2005.

[17] M. Gerig, "Modeling, guidance, and control of aerobatic maneuvers of an autonomous helicopter," Ph.D. dissertation, ETH Zurich, 2008.

[18] H. Khalil, *Nonlinear Systems*. Prentice Hall, 1996.

[19] N. Michael, "Planning and control for teams of robots in complex environments," Ph.D. dissertation, University of Pennsylvania, Philadelphia, PA, Dec. 2008.