Quadrotor Helicopter Trajectory Tracking Control

Gabriel M. Hoffmann*, Steven L. Waslander†
Stanford University, Stanford, California, 94305

Claire J. Tomlin‡
University of California, Berkeley, California, 94720

The Stanford Testbed of Autonomous Rotorcraft for Multi-Agent Control (STARMAC), a fleet of quadrotor helicopters, has been developed as a testbed for novel algorithms that enable autonomous operation of aerial vehicles. This paper develops an autonomous vehicle trajectory tracking algorithm through cluttered environments for the STARMAC platform. A system relying on a single optimization must trade off the complexity of the planned path with the rate of update of the control input. In this paper, a trajectory tracking controller for quadrotor helicopters is developed to decouple the two problems. By accepting as inputs a path of waypoints and desired velocities, the control input can be updated frequently to accurately track the desired path, while the path planning occurs as a separate process on a slower timescale. To enable the use of planning algorithms that do not consider dynamic feasibility or provide feedforward inputs, a computationally efficient algorithm using space-indexed waypoints is presented to modify the speed profile of input paths to guarantee feasibility of the planned trajectory and minimum time traversal of the planned. The algorithm is an efficient alternative to formulating a nonlinear optimization or mixed integer program. Both indoor and outdoor flight test results are presented for path tracking on the STARMAC vehicles.

I. Introduction

The development of autonomous rotorcraft, with the capability to hover, makes possible many applications for uninhabited aerial vehicles (UAVs) for which fixed wing aircraft, with more limited maneuverability, are not well suited. Examples include the close range inspection of buildings and bridges for crack formation, unexploded ordinance detection, rescue beacon tracking of first responders, RFID tracking in commercial package management, and wildlife monitoring. In each case, rapid trajectory generation and precise trajectory tracking are required to enable autonomous rotorcraft operation near obstacles in the environment.

The Stanford Testbed of Autonomous Rotorcraft for Multi-Agent Control (STARMAC) has been developed as a flexible platform upon which novel algorithms can be demonstrated in real world situations with precisely these applications in mind. By developing a multi-vehicle testbed, applications that benefit from multiple simultaneous measurements can be developed, and methods for collision avoidance and cooperative search have already been developed.[1,2] The choice of quadrotor helicopters ensures easy-to-use vehicles with low maintenance requirements, and with a mass of under 3 kg, the testbed can operate in small spaces and cluttered environments with limited safety concerns. Indeed, most of STARMAC flight tests are performed on the Stanford campus, indoors and outdoors, as depicted in Figure 1.

To enable more complex missions for autonomous rotorcraft and for STARMAC in particular, this paper presents a trajectory tracking algorithm to follow a desired path, and an algorithm for the generation of dynamically feasible trajectories. To generate dynamically feasible trajectories, first a plan is generated through the environment which satisfies collision and obstacle avoidance constraints. The planning phase computation is simplified by neglecting vehicle dynamic constraints, making possible real-time planning

*Ph.D. Candidate, Aeronautics and Astronautics; gabeh@stanford.edu. AIAA Student Member.
†Postdoctoral Scholar, Aeronautics and Astronautics; stevenw@stanfordalumni.org. AIAA Member.
‡Professor, Department of Electrical Engineering and Computer Science; tomlin@stanford.edu. AIAA Member.

Figure 1. Autonomous hover at a waypoint by a quadrotor helicopter from the Stanford Testbed of Autonomous Rotorcraft for Multi-Agent Control (STARMAC).

in cluttered environments using techniques such as visibility graphs[3] or fast-marching.[4,5] The resulting trajectories are defined in simple geometric terms of lines and connecting curves with accompanying desired velocities along each segment. Then, a feasible set of inputs and travel speeds is computed, based on the curvature of the path, given speed and acceleration constraints on the vehicles. The ability to hover enables turning with an arbitrary radius of curvature, hence feasible trajectories are guaranteed to exist. Both trajectory tracking algorithms are space-indexed rather than time-indexed, enforcing the requirement that the obstacle-free planned path be tracked without deviation.

The trajectory tracking approach and results demonstrated on the STARMAC platform represent two significant advances for quadrotor vehicles. The first lies in the ability to perform accurate trajectory tracking both indoors and out, and to do so outdoors with all sensing and computation on board the vehicles, as demonstrated by experiments. The second lies in the novel approach to trajectory tracking for quadrotor vehicles, which exploits the hover capabilities of the quadrotor to track paths generated rapidly without consideration of the vehicle dynamics. The space-indexed trajectory that results is minimum-time along the path that it is constrained to follow. Although the resulting travel times through the environment are not necessarily optimal, the lower computational burden of this explicit method ensures that limited on board resources can be used for higher level perception and planning needs. The algorithm is an efficient alternative to alternative solutions. One could formulate a nonlinear program to solve the minimum time space-indexed optimization, requiring an iterative solution, or a mixed integer program to solve the time-indexed optimization, with exponential time complexity in the number of waypoints. The approach presented here requires computational time that is linear in the number of waypoints and does not need to iterate.

The paper proceeds as follows. First, related work in trajectory tracking for helicopters and the development of quadrotor testbeds is described in detail in Section II. Then, a nonlinear dynamic model of the quadrotor vehicle is presented in Section III. Section IV describes the development and flight test results of the quadrotor attitude control as well as the path tracking controller, which demonstrates the capability of the STARMAC platform to perform complex autonomous missions. Sections V then proposes an algorithm for determining a dynamically feasible trajectory given a list of waypoints and desired velocities, and finally, simulation results of the modified trajectories in action are presented in VC.

II. Background

Autonomous trajectory tracking for rotorcraft is a widely studied problem. Recent work has focused primarily on nonlinear methods such as input/output linearization, where differential flatness is used to guarantee that trajectories and control inputs can be generated from an output trajectory,[6] and backstepping controller design[7] which was used to enable acrobatic maneuvers[8] for an X-Cell 0.60 size helicopter. The backstepping controller has been extended to include robustness considerations as well[9] and in each case, flight test results were demonstrated with outdoor testbed vehicles. Although these approaches may be

necessary when extreme performance is required, such as in acrobatic maneuvers, the applications that motivate the development of STARMAC do not place such demands on the trajectory tracking control.

Many research groups are now working on quadrotors as UAV testbeds for control algorithms for autonomous control and sensing.[10–16] A subset of these groups have been successful in performing trajectory tracking flights. One such project features the OS4 vehicle, for which a proportional-derivative (PD) control law led to adequate hovering capability, and an integral backstepping controller for trajectory tracking was shown in simulation.[11] A second project relies on an off-board 100 Hz Vicon state measurement system and LQR control to perform multi-vehicle maneuvers.[17] The vehicles are capable of tracking slow trajectories throughout an enclosed area that is visible to the Vicon system, with limited disruptions. A third project proposed the use of nested saturated control loops for trajectory tracking,[18] for which stability guarantees were established using Lyapunov theory. Experimental demonstrations were performed using the commercially available Draganfly III[19] in tethered indoor flight.

In addition to research platforms, some commercially available small-scale quadrotors are becoming increasingly capable. In particular, the MD4-200 quadrocopter from Microdrones GmbH[20] is a commercially available platform that includes onboard GPS and has is capable of waypoint tracking with 2 m accuracy, outperforming many of the research projects listed above. By contrast, the results presented in this paper demonstrate autonomous path tracking with an indoor accuracy of 10 cm and an outdoor accuracy of 50 cm.

The methodology for trajectory tracking developed for the STARMAC platform strikes a balance between simplicity and performance. It takes as input a path, generated by a higher level motion planner which is simplified by not needing to consider dynamic constraints, and produces a dynamically feasible sub-optimal trajectory that the vehicle tracks using a linear controller. Although this approach is conservative in the type of paths it generates through environments, it provides sufficient capabilities for the tasks required of the quadrotor testbed in most applications. The flight test results demonstrate a significant improvement in capability over previous quadrotor testbeds.

III. Vehicle Dynamics

The nonlinear dynamics of the quadrotor helicopter are those of a point mass m with moment of inertia $I_b \in \mathbb{R}^{3\times3}$, location $\rho \in \mathbb{R}^3$ in inertial space, and angular velocity $\omega_{\mathbf{B}} \in \mathbb{R}^3$ in the body frame. The vehicle undergoes forces $\mathbf{F} \in \mathbb{R}^3$ in the inertial frame and moments $\mathbf{M} \in \mathbb{R}^3$ in the body frame[a] yielding the equations of motion,

$$\mathbf{F} = m\ddot{\rho} \tag{1}$$

$$\mathbf{M} = I_b\dot{\omega}_{\mathbf{B}} + \omega_{\mathbf{B}} \times I_b\omega_{\mathbf{B}} \tag{2}$$

where the total angular momentum of the rotors is assumed to be near zero, as the momentum from the counter-rotating pairs cancels when yaw is held steady, and is small in practice. The body attitude is represented as a 1-2-3 Euler angle sequence with roll ϕ, pitch θ, and yaw ψ. A free body diagram is presented in Figure 2, depicting the main forces and moments acting on the vehicle, including the motor thrusts and torques, gravity and aerodynamic drag.

The i^{th} rotor contributes thrust force T_i, along local axis $\mathbf{z}_{R_i}$ perpendicular to the rotor plane[b], which may be tilted with respect to the body frame due to blade flapping, an aerodynamic effect described in detail in previous work.[21] The thrust acts at a distance l from the center of gravity, denoted $\mathbf{l}_i$ in vector form. Each rotor produces moments $\mathbf{M}_i$ in the body fixed frame. The moment is a function of the motor torque, thrust, and additional aerodynamic effects. Note that for the design of the trajectory controller, it is assumed that the disturbances in forces and moments from aerodynamic effects due to vehicle velocity can be satisfactorily rejected in the attitude and attitude feedback control loops presented in Section IVA, and can therefore be neglected.

The rotor thrust and reaction torque are proportional to the rotor rotation speed squared, which can be linearized for control.[21] Roll and pitch are controlled through torque generated by differential speed of opposing pairs of rotors; rotors 2 and 4 for roll, and rotors 1 and 3 for pitch, as shown in Figure 3a. This yields approximately independent moments about each axis. The yaw angle is controlled through torque

[a]The subscript I shall denote inertial axes in North, East, Down coordinates, with labels $\mathbf{e_N}, \mathbf{e_E}, \mathbf{e_D}$. The subscript B shall denote the body reference frame, with coordinates $\mathbf{x_B}, \mathbf{y_B}, \mathbf{z_B}$.

[b]The subscript R_i denotes to i^{th} rotor plane reference frame.

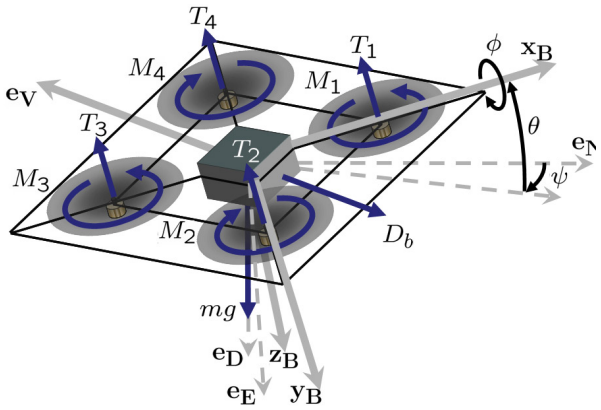
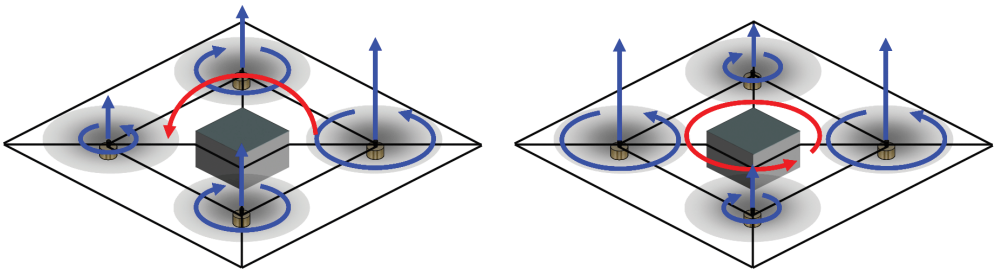


Figure 2. Free body diagram of a quadrotor helicopter. The roll, pitch and yaw angles are denoted ϕ, θ, and ψ, respectively. Motors are labeled $1-4$, with thrust T_i and moments M_i. The force due to gravity, mg, acts in the downward direction and drag forces D_b act opposite the direction of the free stream velocity $\mathbf{e_V}$.



(a) (b)

Figure 3. Quadrotor helicopters are controlled by varying the speeds of the rotors. (a) Pitch and roll torques are generated independently by controlling the relative speed of rotors on opposite sides of the vehicle. (b) Yaw torque is generated by controlling the relative speed of pairs of motors, which counter-rotate to produce a varying total reaction torque about the motor shafts. Vertical acceleration is controlled by the total speed of all rotors, and lateral acceleration is controlled through the pitch and roll of the aircraft.

generated by differential speed between rotors 1 and 3, which rotate one direction, and rotors 2 and 4, which rotate the opposite direction, as shown in Figure 3b. This varies the total reaction torque about the yaw axis without affecting the roll or pitch moments, or total thrust.

The vehicle body drag force D_b is a function of both vehicle and wind speed, and acceleration due to gravity is denoted g. The total force $\mathbf{F}$ is the summation of all contributing forces,

$$\mathbf{F} = -D_b \mathbf{e_v} + mg\mathbf{e_D} + \sum_{i=1}^{4} (-T_i R_{R_i,I} \mathbf{z}_{R_i}) \tag{3}$$

where $R_{R_i,I}$ is the rotation matrix from the plane of rotor i to inertial coordinates[c]. Similarly, the total moment, $\mathbf{M}$, is,

$$\mathbf{M} = \sum_{i=1}^{4} (\mathbf{M}_i + \mathbf{l}_i \times (-T_i R_{R_i,B} \mathbf{z}_{R_i})) \tag{4}$$

where $R_{R_i,B}$ is the rotation matrix from the plane of rotor i to body coordinates. Note that the drag force was neglected in computing the moments acting on the vehicle, as it was found to cause a negligible disturbance on the total moment over the flight regime of interest compared to other aerodynamic moments. Combining Equations (2) with (3) and (4) yields a full set of nonlinear dynamics for the quadrotor vehicles.

[c]The notation $R_{A,B}$ shall refer to rotation matrices from coordinate system A to B throughout.

IV. Trajectory Tracking Control

The proposed control law tracks line segments connecting sequences of waypoints at a desired velocity. To design the trajectory tracking control law, the strong attitude angle control authority of quadrotor helicopters can be exploited. This section proceeds by first presenting an attitude control law with the ability to track rapidly varying commands. The altitude control law is given in previous work, using feedback linearization and acceleration feedback.[21] Next, the path tracking control law is presented, which uses the attitude controller to orient the vehicle's thrust to produce the desired lateral acceleration.

A. Attitude Control

The equations of motion are approximately decoupled about each attitude axis, so control inputs about each axis, u_ϕ, u_θ, and u_ψ, can be implemented independently. This is an accurate assumption when the angular velocities are low and the attitude angles are within approximately $\pm 30^\circ$. The inputs for each axis can then be combined to generate u_1 through u_4, for motors 1 through 4,

$$\begin{aligned} u_1 &= -u_\theta + u_\psi + u_z \\ u_2 &= u_\phi - u_\psi + u_z \\ u_3 &= u_\theta + u_\psi + u_z \\ u_4 &= -u_\phi - u_\psi + u_z \end{aligned} \tag{5}$$

In controlling each attitude angle, the time delay in thrust must be included in the model. It is well approximated as a first order delay, as experimentally verified.[21] Linearizing Equation (4), the resulting transfer function for the roll axis is

$$\frac{\Phi(s)}{U_\phi(s)} = \frac{I_\phi / l}{s^2(\tau s + 1)} \tag{6}$$

where I_ϕ is the component of I_b for the roll axis, l is once again the lever arm of the rotors about the center of mass, and τ is the thrust time delay, which was experimentally determine to be 0.1 sec for STARMAC. The transfer functions for the pitch and yaw axes are similarly computed.

Although a standard proportional-integral-derivative (PID) controller has been shown to perform adequately,[22] control design using root locus techniques revealed that adding an additional zero, using angular acceleration feedback, allowed the gains to be significantly increased, yielding higher bandwidth. Also note that aerodynamic disturbance torques are directly measured by errors in the angular acceleration. The resulting control law,

$$C(s) = k_{dd} s^2 + k_d s + k_p + \frac{k_i}{s} \tag{7}$$

was tuned to provide substantially faster and more accurate performance than previously possible. The Laplace transform of the control input is then,

$$U(s) = C(s)(\Theta_{ref}(s) - \Theta(s)) \tag{8}$$

The implementation of this control law requires some consideration. First, the required angular acceleration signal must be computed by finite differencing the rate gyroscope data, a step which can amplify noise. However, in implementation, it was found that the signal resulting from a single difference using 76 Hz measurements from rate gyroscope chips was sufficiently clean for use in the controller. Second, to implement this control law, $C(s)$ acts on the error signal to provide *reference command tracking* capability. To enable this, it was necessary to process the reference command using a low pass digital filter to compute both first and second derivatives.

In practice, the controller is able to track rapidly varying reference commands, as shown in Figure 4, with root mean square (RMS) error of 0.65° in each axis. Aggressive flights have been flown frequently, with typically up to 15° of bank angle. The controller has been flown up to its programmed limit of 30° without apparent degradation in performance.

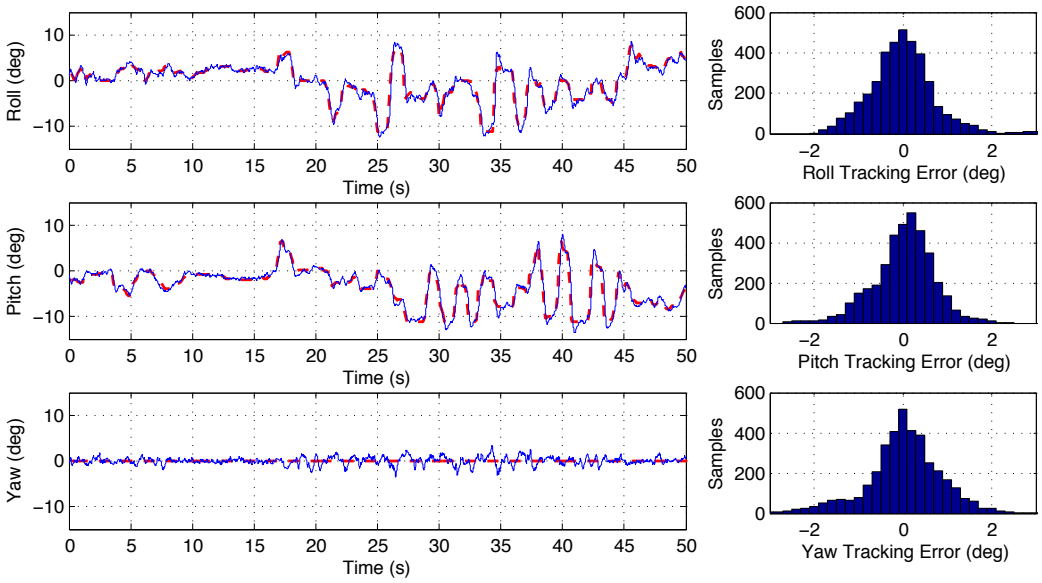


Figure 4. Attitude control results from a flight test a STARMAC quadrotor, accurately tracking a time varying reference command. The dashed line is the reference command, the solid line is the measured vehicle attitude angle.

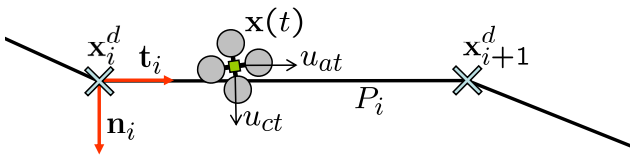


Figure 5. Vehicle path definition. While the quadrotor travels along segment P_i from waypoint $\mathbf{x}_i^d$ to $\mathbf{x}_{i+1}^d$, it applies along track and cross track control inputs u_{at} and u_{ct} to follow the path segment.

B. Path Tracking Control

With an attitude controller of sufficient bandwidth defined, it is possible to develop a path tracker that generates attitude reference commands, ignoring the fast dynamics of the attitude angles.

A path, $P \in N \times \mathbb{R}^3$, is defined by a sequence of N desired waypoints, $\mathbf{x}_i^d$ and desired speeds of travel v_i^d along path segment P_i connecting waypoint i to $i+1$, as depicted in Figure 5. Let $\mathbf{t}_i$ be the unit tangent vector in the direction of travel along the track from $\mathbf{x}_i^d$ to $\mathbf{x}_{i+1}^d$, and $\mathbf{n}_i$ be the unit normal vector to the track. Then, given the current position $\mathbf{x}(t)$, the cross track error e_{ct} and along track error rate $\dot{e}_{at}$ are,

$$\begin{aligned} e_{ct} &= (\mathbf{x}_i^d - \mathbf{x}(t)) \cdot \mathbf{n}_i \\ \dot{e}_{ct} &= -\mathbf{v}(t) \cdot \mathbf{n}_i \\ \dot{e}_{at} &= v_i^d - \mathbf{v}(t) \cdot \mathbf{t}_i \end{aligned} \tag{9}$$

Note that only the along track error rate is considered, and depends only the velocity of the vehicle. This is done so that the resulting controller does not attempt to catch up or slow down for scheduled waypoints, but simply proceeds along the track matching the desired velocity as closely as possible.

A trajectory tracking controller was implemented by closing the loop on along track velocity and cross track error. This is essentially piecewise PI control in the along track direction, and PID control in the cross

track direction.

$$\begin{aligned} u_{at} &= K_{atp}\dot{e}_{at} + K_{ati}\int_0^t \dot{e}_{at}dt \\ u_{ct} &= K_{ctp}e_{ct} + K_{ctd}\dot{e}_{ct} + K_{cti}\int_0^t \dot{e}_{ct}dt \end{aligned} \tag{10}$$

Transition from segment i to $i+1$ occurs when the vehicle crosses the line segment normal to the path at the end of the segment. Upon completion of P_i, the integrators were reset because the along track and cross track error integrals reject different disturbances. Note than when the transition angle from one segment to the next is sufficiently small, as occurs for low curvature paths, the integrators need not be reset.

The controller defined in Equation 7 was implemented on the STARMAC platform in both indoor and outdoor settings, for which results are presented in Figures 6 and 7, respectively. The indoor results demonstrate tracking errors of under 10cm throughout the box shaped trajectory, and show the largest overshoot when switching from one track to the next, as the desired direction of travel suddenly switches by 90°. For the outdoor flight tests, the gains on the cross track and along track controllers were reduced significantly, and the resulting errors increased to ±0.5 m. Lower gains were used due to increased oscillations when in hover condition outside, and may be attributed to either significant levels of wind gust disturbances or to the decreased position update rate from 15 Hz for the indoor positioning system to 10 Hz for the carrier phase differential GPS solution. Further investigation is required to isolate the true source of this drop in performance.

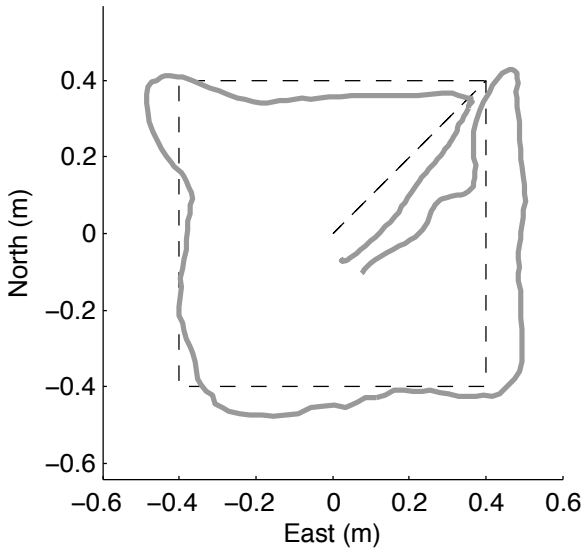


Figure 6. Tracking a trajectory indoors, at 0.5 m/s, with an error of under 0.1 m. An overhead webcam is used to provide 15 Hz position updates simulating GPS data.

One thing to note in the above flight test results, in particular in the case of the indoor trajectory, is the effect of direction changes in the trajectory to be tracked. Since the trajectory controller as defined in Equation 10 has no sense of the curvature of the desired trajectory, significant overshoot on sharp corners is inevitable. This can cause concern in constrained environments if, for example the changes in direction are required to avoid obstacles or other vehicles in the environment. Section V now presents modifications to the trajectory to be tracked that do incorporate path curvature and vehicle dynamics when making transitions from one waypoint to the next, enabling more precise tracking performance.

V. Time Optimal Feasible Trajectory

Given a planned sequence of desired waypoints, the desired speed profile for the quadrotor must be generated in a dynamically feasible manner. Although the speed profile can be determined at the time of

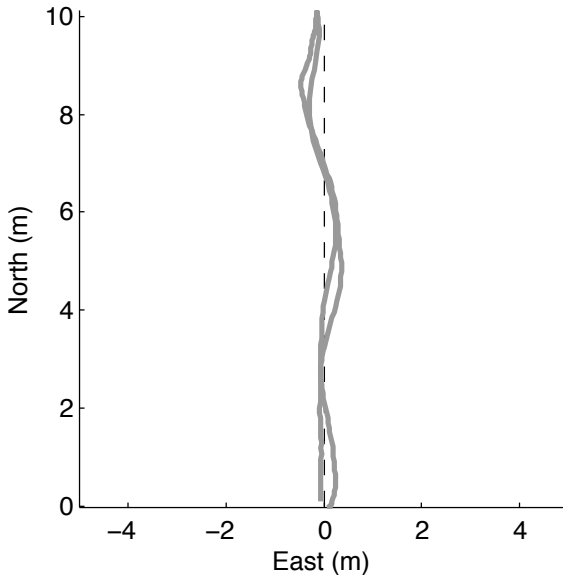


Figure 7. Tracking a trajectory outdoors, at 2.0 m/s, with an error of under 0.5 m. GPS is used for position, and nominal winds were measured at 5-10 mph.

trajectory generation, this step increases the complexity of trajectory planning and many algorithms with low complexity can quickly generate paths, but cannot guarantee dynamic feasibility. Examples include using A^* with visibility graphs,[23] and fast marching[5] with gradient descent, as shown in Figure 8. In order to render the trajectory dynamically feasible, this section presents a method to compute a space-indexed speed profile that traverses the sequence of waypoints in minimum time while satisfying acceleration and speed constraints. This algorithm runs in $\mathcal{O}(N)$ time, where N is the number of waypoints.

This method starts by generating a finer sampling of waypoints than the examples in Section B, which defines a space-indexed path, $\mathbf{x} \in \mathbb{R}^{N\times 2}$, with a curvature that can be numerically estimated[d]. For computational efficiency, the algorithm assumes acceleration constraints apply separately in the along track and cross track directions. First, the cross track acceleration constraint is satisfied by computing a maximum allowable speed at every waypoint, using the approximate path curvature. Second, the optimal time speed profile is computed that satisfies piecewise linear along track acceleration constraints, as well as the minimum of the desired speed and of the allowable speed from the previous step.

A. Cross Track Acceleration Constraint Satisfaction

The cross track acceleration $a_{ct,i}$ at waypoint $\mathbf{x}_i = (x_i, y_i)$ is a function of the velocity $v_{ct,i}$ at $\mathbf{x}_i$ and the radius of curvature r_i,

$$a_{ct,i} = \frac{v_i^2}{r_i} \tag{11}$$

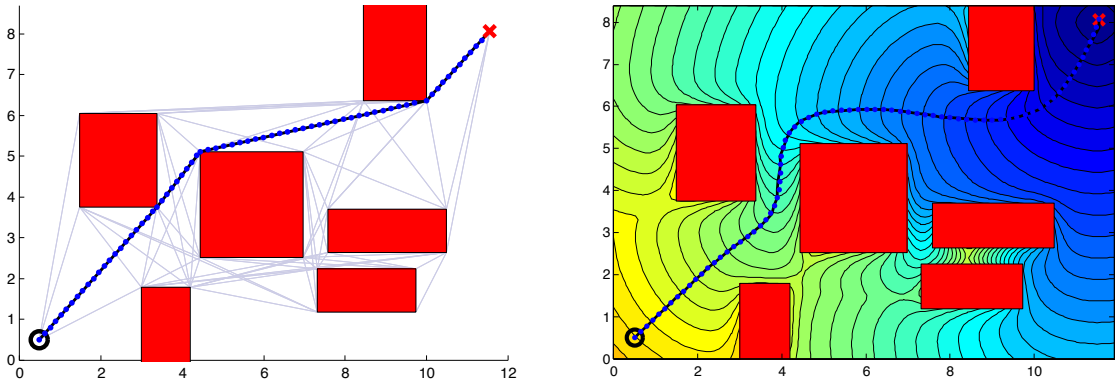
Constraining the cross track acceleration to be of magnitude less than a_{max} results in a maximum allowable velocity $v_{i,allow}$ at $\mathbf{x}_i$,

$$v_{i,allow} \leq \sqrt{a_{max} r_i} \tag{12}$$

To determine r_i, waypoints $\mathbf{x_{i-1}}$, $\mathbf{x}_i$, and $\mathbf{x_{i+1}}$ are used to define a circle through those points. This circle is assumed to have curvature r_i, a valid assumption when the the waypoints are finely spaced relative to their curvature. Waypoints $\mathbf{x_{i-1}}$ and $\mathbf{x_i}$ define a first line, and points $\mathbf{x_i}$ and $\mathbf{x_{i+1}}$ define a second line, with slopes m_a and m_b respectively. The intersection of the perpendicular lines through their midpoints is at the center of the circle. The slopes are

$$m_a = \frac{y_i - y_{i-1}}{x_i - x_{i-1}}, \qquad m_b = \frac{y_{i+1} - y_i}{x_{i+1} - x_i} \tag{13}$$

[d]Note that the variable $\mathbf{x}$ is reused by abuse of notation, in order to greatly simplify the derivation in this section.



(a) (b)

Figure 8. (a) Visibility Graph, with shortest path highlighted, and waypoints sampled with a spacing of 0.2 units. (b) Fast marching potential function with steepest descent path highlighted.

Solving for the intersection yields the center point x_i^c,

$$x_i^c = \frac{m_a m_b(y_{i-1} - y_{i+1}) + m_b(x_{i-1} + x_i) - m_a(x_i + x_{i+1})}{2(m_b - m_a)} \tag{14}$$

This can be substituted into the equation for the line to obtain y_i^c. Equation (13) can be substituted into (14) to obtain a form that is numerically well behaved. This yields

$$x_i^c = \frac{(x_{i-1}^2 + y_{i-1}^2)(y_i - y_{i+1}) + (x_i^2 + y_i^2)(y_{i+1} - y_{i-1}) + (x_{i+1}^2 + y_{i+1}^2)(y_{i-1} - y_i)}{c_i} \tag{15}$$

$$y_i^c = \frac{(x_{i-1}^2 + y_{i-1}^2)(x_{i+1} - x_i) + (x_i^2 + y_i^2)(x_{i-1} - x_{i+1}) + (x_{i+1}^2 + y_{i+1}^2)(x_i - x_{i-1})}{c_i} \tag{16}$$

where

$$c_i = 2(y_{i+1}(x_i - x_{i-1}) + y_i(x_{i-1} - x_{i+1}) + y_{i-1}(x_{i+1} - x_i)) \tag{17}$$

Note that c_i should be precomputed as it's value is zero for any straight line segment where the radius of curvature is infinite. Otherwise, the centerpoint can be computed using (15). The radius r_i is the norm of the vector connecting $\mathbf{x}_i$ and the center. The direction of the curve can be computed for finite r_i using the sign of the cross product of the first and second line segments. Now, with a speed limit imposed by acceleration constraints, $v_{i,allow}$, at every waypoint, the speed plan and feedforward acceleration can be computed for the path, as presented next.

B. Time Optimal Speed and Acceleration Plan

The speed at every point is constrained by both the desired velocity $v_{i,max}$ as defined by the pre-path, which must be feasible for the vehicle to fly, and by the constraint imposed for the cross track acceleration constraint, $v_{i,allow}$. At every point, the minimum of these two constraints must be satisfied. Let the maximum velocity allowed at each space-indexed point be,

$$\bar{v}_i = min(v_{i,max}, v_{i,allow}) \tag{18}$$

This section derives the algorithm to determine the speed profile, $\mathbf{v} \in \mathbb{R}^N$, and acceleration profiles, $\mathbf{a_{at}} \in \mathbb{R}^N$ and $\mathbf{a_{ct}} \in \mathbb{R}^N$, that satisfy these constraints.

Consider piecewise constant along track acceleration $a_{at,i}$ between waypoints i and $i+1$. The speed at the next waypoint v_{i+1} given current speed v_i and time between waypoints Δt is

$$v_{i+1} = v_i + a_{at,i}\Delta t \tag{19}$$

Let the along track position of waypoint i be s_i, measured from the beginning of the path, i.e., $||\mathbf{x}_{i+1}-\mathbf{x}_i||_2 = s_{i+1} - s_i$. To determine Δt, integrate (19).

$$\frac{1}{2}a_{at,i}\Delta t^2 + v_i\Delta t + (s_i - s_{i+1}) = 0 \tag{20}$$

Solving for Δt yields

$$\Delta t = \frac{-v_i \pm \sqrt{v_i^2 - 2a_{at,i}(s_i - s_{i+1})}}{a_{at,i}} \tag{21}$$

Substituting (21) into (19),

$$v_{i+1} = \sqrt{v_i^2 - 2a_{at,i}(s_i - s_{i+1})} \tag{22}$$

Note that only the positive root in (21) has a physical interpretation because the speeds must be positive. Solving for the acceleration to achieve a given change in speed over a given distance,

$$a_{at,i} = \frac{v_i^2 - v_{i+1}^2}{2(s_i - s_{i+1})} \tag{23}$$

Algorithm 1 Velocity_Plan_Sweep(v_0, $\mathbf{s}$, $\bar{\mathbf{v}}$, a_{max})

$v_1 \leftarrow v_0$
for $i = 1$ to N **do**
  $a_{at,i} \leftarrow \min\left(a_{max}, \frac{v_i^2-(\bar{v}_{i+1})^2}{2(s_i-s_{i+1})}\right)$
  if $a_{at,i} > 0$ **then**
    $v_{i+1} \leftarrow \sqrt{v_i^2 - 2a_{at,i}(s_i - s_{i+1})}$
    $\Delta t_i \leftarrow \frac{-v_i+\sqrt{v_i^2-2a_{at,i}(s_i-s_{i+1})}}{a_{at,i}}$
  else
    if $v_i > \bar{v}_{i+1}$ **then**
      $v_{i+1} \leftarrow \bar{v}_{i+1}$
      $a_{at,i} \leftarrow \frac{v_i^2-v_{i+1}^2}{2(s_i-s_{i+1})}$
      $\Delta t_i \leftarrow \frac{-v_i+\sqrt{(v_i^2-2a_{at,i}(s_i-s_{i+1}))}}{a_{at,i}}$
    else
      $v_{i+1} \leftarrow v_i$
      $a_{at,i} \leftarrow 0$
      $\Delta t_i \leftarrow \frac{s_{i+1}-s_i}{v_i}$
    end if
  end if
end for
return $\mathbf{v}$, $\mathbf{a}$, $\mathbf{\Delta t}$

Next, the algorithm to compute the time optimal plan is explained. It operates in five steps. First, all along track positions are stored in $\mathbf{s}$ and all maximum speeds are stored in $\bar{\mathbf{v}}$. Second, to prepare for sweeping through the points from the end to the beginning, these arrays are reversed in time, and $\mathbf{s}$ is negated. Third, to satisfy the minimum acceleration constraint the sweeping algorithm, defined in Algorithm 1 and described below, is run on the reversed data with $v_0 = \bar{v}_N$ to generate an along track acceleration plan, $\mathbf{a}$, such that $a_{at,i} > -a_{max}\ \forall\ i$. Third, to retain the minimum acceleration constraint from the previous step, $\bar{\mathbf{v}}$ is set to the resulting $\mathbf{v}$. Fourth, to prepare for sweeping through the points from beginning to end, the arrays are again reversed in time, and $\mathbf{s}$ is negated. Fifth, to enforce the maximum acceleration constraint, the sweeping algorithm is run on this data, with $v_0 = \bar{v}_1$, to generate $\mathbf{a}$ such that $a_{at,i} < a_{max}\ \forall\ i$. In this sweep, $a_{at,i} < -a_{max}$ is not possible due to the speed limits imposed by the previous, reverse sweep. Once these reverse and forward sweeps have been completed, the acceleration constraints and speed constraints are satisfied.

The sweeping algorithm, Algorithm 1, operates by incrementing through the waypoints. At waypoint i, it first determines if it is possible to accelerate between $\mathbf{x}_i$ and $\mathbf{x}_{i+1}$. If it can accelerate, it accelerates as fast

as required, or saturates at a_{max}. If it cannot accelerate, it next checks if it must slow down to satisfy $\bar{v}_{i+1}$. If it must, it slows down to that speed, using whatever acceleration necessary, even if using $a_{at,i} < -a_{max}$ if required. Note that it only violates a_{max} to *slow down*. If it does not need to slow down, then it holds the current speed. Finally, it returns the computed speed profile, $\mathbf{v}$, acceleration profile, $\mathbf{a}$, and time profile, Δt. Note that Δt must be computed, because the algorithm is *space-indexed*.

Using this algorithm, the time optimal speed and along track acceleration can be computed. The cross track acceleration required to achieve this path has magnitude v_i/r_i^2, with the sign equal to that of the third component of the cross product of tangent vectors $\mathbf{t}_{i-1}$ and $\mathbf{t}_i$. The computed feedforward accelerations can then be superimposed with the feedback control law, (10), to control the quadrotor to track the desired trajectory at the time-optimal speed.

C. Application to Example Problems

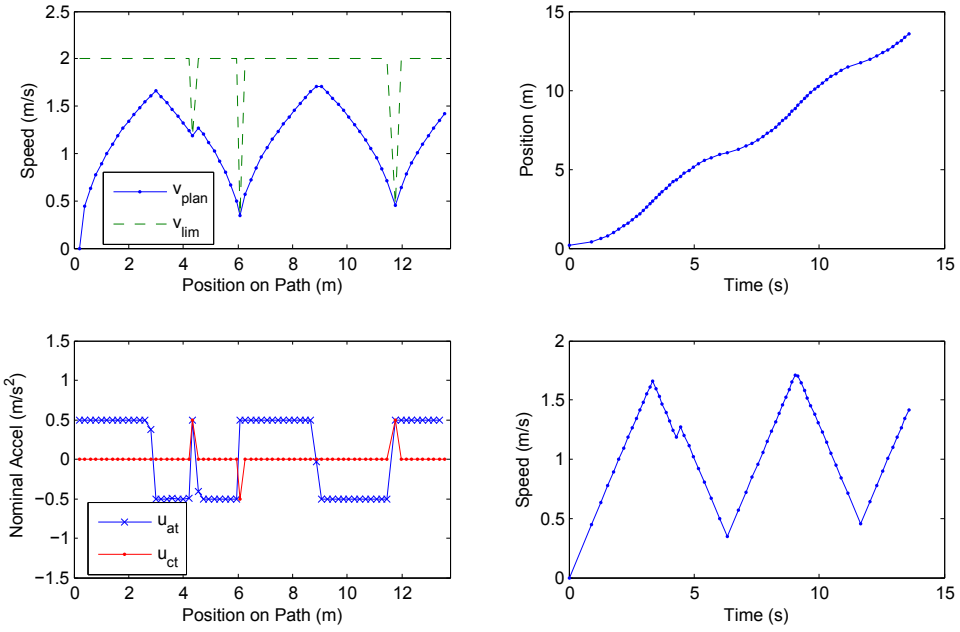


Figure 9. Results from optimizing the visibility graph based path, yielding a 13.6 sec plan. When the path changes direction around obstacle corners, the vehicle must slow down to a safe speed, accelerate laterally, then speed up again.

The time optimal sweeping algorithm was applied to the two examples of path planning algorithms given in the beginning of this section, and shown in Figure 8. All experiments were run using a 3.4 GHz Pentium IV computer and the time to compute the plan demonstrated that this algorithm is negligible, as expected by its $\mathcal{O}(n)$ time complexity. The plans demonstrate the ability of the algorithm to capture the necessary information to produce a time-optimal trajectory.

1. *Visibility Graph*

Consider the path generated using a A* to find the shortest path through a visibility graph,[23] shown in Figure 8a. The generation of the shortest path required under 20 ms using Matlab. However, this method ignored dynamic constraints, which would require the quadrotor to slow to a halt at sharp corners. To run the optimum time sweeping algorithm, the shortest path was discretized at 0.2 m increments, resulting in a computation time of under 1 ms to run optimum time sweeping. The results are shown in Figure 9. The minimum time to traverse the path under acceleration and speed constraints was 13.6 sec.

One drawback to the presented approach is that corners may be cut, depending on the fineness of the discretization, a problem common to many path planning algorithms. However, the computational cost for

choosing a sufficiently fine discretization is low, given the low run time of the sweeping algorithm and so an arbitrary precision may be used to avoid obstacles. The effect of increasing the precision of the space discretization is that it increases the total time to complete the trajectory, as the velocity slows to zero at each turn in the underlying desired path. In some problem formulations, it is desirable for corners to be cut, if there is sufficient margin in the definition of obstacles used in generating the original plan. This allows sharp corners in the paths to be traversed without stopping, which is only advantageous when there are sharp corners in the path as in the case of the visibility graph. Fast marching, on the other hand, does not suffer from this trade off.

2. *Fast Marching*

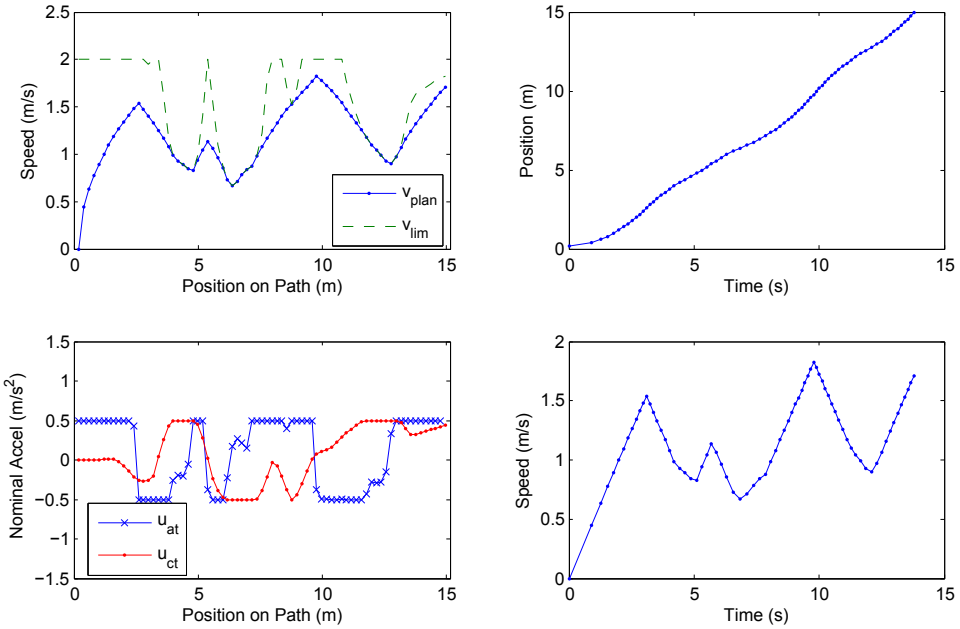


Figure 10. Results from optimizing the fast marching based path, yielding a 13.8 sec plan. Although this path is longer than the visibility graph based path, there are no corners, so the average speed is faster.

Consider the path generated using fast marching with the planar wave approximation,[5] shown in Figure 8b. This method, using a discrete grid, propagates the cost to reach the goal from the goal outward, increasing with distance. Close proximity to obstacles increases this cost, resulting in the refraction of the wave front around surfaces. Using the resulting cost map, gradient descent is used to find the best path to the target, according to the cost map. In the example shown, the complete path finding algorithm again ran in under 20 ms. Due to the aversion to obstacles, this cost map is not exactly the time to the goal–the vehicle is penalized for being close to obstacles. To run the optimum time sweeping algorithm, the fast marching path was discretized at 0.2 m increments again, and resulted in a computation time of under 1 ms. The results are shown in Figure 10. The minimum time to traverse the path under acceleration and speed constraints was 13.8 sec.

The results for fast marching differ from those of the visibility graph. Although the aversion to obstacles results in smoother paths, the paths are always longer than the visibility graph. As a result, sometimes the visibility graph is faster, and sometimes the fast marching path is faster. The aversion to obstacles causes fast marching to tend to avoid going through small gaps when possible, which is advantageous if there are winding corners, but hurts performance if there is a simple gap. The resulting minimum time acceleration commands for the fast marching path varied much more smoothly than for the visibility graph, due to a lack of corners. Although the maximum velocity constraint due to cross track acceleration was active for more time than it was for the visibility graph path, the minimum values were not as low, so the average speeds

were close to equal for the two examples.

In both examples, the computation time to find a path through the environment was low because dynamic constraints were first ignored to find the path, and then included by the optimum time sweeping algorithm. By separately considering vehicle dynamics and path plans, the overall paths are not optimal. However, given the already good paths, the optimum time sweeping algorithm generates a control input and speed plan that traverses the path in minimum time. This algorithm can be run online in real-time with the feedback controller of Section B.

VI. Conclusions

Autonomous rotorcraft can make possible many potential applications for uninhabited aerial vehicles. To enable more complex missions for autonomous quadrotors, and for STARMAC in particular, this paper presented a trajectory tracking algorithm to follow a desired path, and an algorithm for the generation of dynamically feasible trajectories. The trajectory tracking algorithm used has been experimentally demonstrated to track a path indoors with 10 cm accuracy and outdoors with 50 cm accuracy. The dynamically feasible trajectory generation algorithm separated the plan generation for obstacle avoidance from the computation of dynamically feasible travel speeds and control inputs. In doing so, it is possible to find the time optimal inputs to follow a given path with little computational burden. Simulations demonstrate the accurate computation of these control inputs, and the fast run time on the computer.

In future work, it would be interesting to investigate the possibility of generating trajectories that slightly modify the location of space indexed waypoints to improve the minimum travel speed for the worst offending segments along the path. It may also be possible to incorporate knowledge of the obstacle locations in the velocity determination to ensure slower, more precise tracking when in constrained spaces. The proposed algorithm will be next be implemented and tested on the STARMAC platform, and flight test results from these experiments will be included in the final version of the paper. With reliable tracking of trajectories achieved, the STARMAC platform will be ready to embark on many of the applications envisaged for it, including multi-vehicle collision avoidance and cooperative target localization in unknown environments.

Acknowledgments

The authors would like to thank Vijay Pradeep, Haomiao Huang and Michael Vitus for their many contributions to the STARMAC project, and in particular, Vijay Pradeep, for his work on the fast-marching algorithm.

References

[1]Hoffmann, G. M., Waslander, S. L., and Tomlin, C. J., "Distributed Cooperative Search using Information-Theoretic Costs for Particle Filters with Quadrotor Applications," *Proceedings of the AIAA Guidance, Navigation, and Control Conference*, Keystone, CO, August 2006.

[2]Waslander, S. L., *Multi-agent system design for aerospace applications*, Ph.D. thesis, Stanford University, Stanford, CA, USA, 2007.

[3]Asand, T., Asano, T., Guibas, L., Hershberger, J., and Imai, H., "Visibility of Disjoint Polygons," *Algorithmica*, Vol. 1, 1986, pp. 49–63.

[4]Kimmel, R. and Sethian, J., "Computing geodesic paths on manifolds," *Proceedings of National Academy of Sciences*, Vol. 95, 1998, pp. 8431–8435.

[5]Osher, S. and Fedkiw, R., *Level Set Methods and Dynamic Implicit Surfaces*, Springer-Verlag, New York, NY, USA, 2002.

[6]Koo, T. J. and Sastry, S., "Output Tracking Control Design of a Helicopter Model Based on Approximate Linearization," *Proceedings of the 37th IEEE Conference on Decision and Control*, Tampa Bay, Florida, USA, December 1998.

[7]Frazzoli, E., Dahleh, M., and Feron, E., "Trajectory tracking control design for autonomous helicopters using a backstepping algorithm," *AACC American Control Conference*, 2000, pp. 4102 – 4107.

[8]Gavrilets, V., Martinos, I., Mettler, B., and Feron, E., "Flight test and simulation results for an autonomous aerobatic helicopter," Vol. 2, 2002, pp. 8C3–1–8C3–6, DOI: 10.1109/DASC.2002.1052943.

[9]Mahony, R. and Hamel, T., "Robust trajectory tracking for a scale model autonomous helicopter," *International Journal of Robust and Nonlinear Control*, Vol. 14, No. 12, 2004, pp. 1035–1059, DOI: 10.1002/rnc.931.

[10]Altuğ, E., Ostrowski, J. P., and Taylor, C. J., "Quadrotor Control Using Dual Camera Visual Feedback," *In Proceedings of the IEEE International Conference on Robotics and Automation*, Taipei, Taiwan, Sept 2003, pp. 4294–4299.

[11]Bouabdallah, S., Murrieri, P., and Siegwart, R., "Towards Autonomous Indoor Micro VTOL," *Autonomous Robots*, Vol. 18, No. 2, March 2005, pp. 171–183.

[12]Guenard, N., Hamel, T., and Moreau, V., "Dynamic modeling and intuitive control strategy for an X4-flyer," *In Proceedings of the International Conference on Control and Automation*, Budapest, Hungary, June 2005, pp. 141–146.

[13]Escareño, J., Salazar-Cruz, S., and Lozano, R., "Embedded control of a four-rotor UAV," *Proceedings of the AACC American Control Conference*, Minneapolis, MN, June 2006, pp. 3936–3941.

[14]Nice, E. B., *Design of a Four Rotor Hovering Vehicle*, Master's thesis, Cornell University, 2004.

[15]Park, S., Won, D., Kang, M., Kim, T., Lee, H., and Kwon, S., "RIC (Robust Internal-loop Compensator) Based Flight Control of a Quad-Rotor Type UAV," *Proceedings of the IEEE/RSJ International Conference on Intelligent Robotics and Systems*, Edmonton, Alberta, August 2005.

[16]Pounds, P., Mahony, R., and Corke, P., "Modelling and Control of a Quad-Rotor Robot," *In Proceedings of the Australasian Conference on Robotics and Automation*, Auckland, New Zealand, 2006.

[17]Culligan, K., Valenti, M., Kuwata, Y., and How, J. P., "Three-Dimensional Flight Experiments Using On-Line Mixed-Integer Linear Programming Trajectory Optimization," *Proceedings of the AACC American Control Conference*, New York, NY, July 2006, pp. 5322–5327.

[18]Salazar-Cruz, S., Palomino, S., and Lozano, R., "Trajectory tracking for a four rotor mini-aircraft," *Proceedings of the IEEE Conference on Decision and Control*, Seville, Spain, December 2005, pp. 2505–2510.

[19]"DraganFly-Innovations DraganFlyer III," 2006, http://www.rctoys.com.

[20]*Microdrones GmbH MD4-200 quadrotor helicopter*, 2008, http://www.microdrones.com/news_waypoint_navigation.html.

[21]Hoffmann, G. M., Huang, H., Waslander, S. L., and Tomlin, C. J., "Quadrotor Helicopter Flight Dynamics and Control: Theory and Experiment," *Proceedings of the AIAA Guidance, Navigation, and Control Conference*, Hilton Head, SC, August 2007.

[22]Waslander, S. L., Hoffmann, G. M., Jang, J. S., and Tomlin, C. J., "Multi-Agent Quadrotor Testbed Control Design: Integral Sliding Mode vs. Reinforcement Learning," *In Proceedings of the IEEE/RSJ International Conference on Intelligent Robotics and Systems 2005*, Edmonton, Alberta, August 2005, pp. 468–473.

[23]Latombe, J.-C., *Robot Motion Planning*, Kluwer Academic Publishers, Boston, MA, USA, 1991.