1

Trajectory Generation and Control for Precise Aggressive Maneuvers with Quadrotors

Daniel Mellinger, Nathan Michael, Vijay Kumar

GRASP Laboratory,
University of Pennsylvania,
Philadelphia, PA 19104, USA
{dmel, nmichael, kumar}@seas.upenn.edu

Summary. We study the problem of designing dynamically feasible trajectories and controllers that drive a quadrotor to a desired state in state space. We focus on the development of a family of trajectories defined as a sequence of segments, each with a controller parameterized by a goal state. Each controller is developed from the dynamic model of the robot and then iteratively refined through successive experimental trials to account for errors in the dynamic model and noise in the actuators and sensors. We show that this approach permits the development of trajectories and controllers enabling aggressive maneuvers such as flying through narrow, vertical gaps and perching on inverted surfaces with high precision and repeatability.

1.1 Introduction

In this paper we study the problem of designing dynamically feasible trajectories and controllers that drive a quadrotor to a desired state (position, orientation, linear and angular velocity) in state space. We focus on the development of a family of trajectories defined as a sequence of segments, each with a controller parameterized by a goal state. Each controller is developed from the dynamic model of the robot, and then iteratively refined through successive experimental trials to account for errors in the dynamic model and noise in the actuators and sensors. We show that this approach permits the development of trajectories and controllers enabling aggressive maneuvers such as flying through narrow, vertical gaps and perching on inverted surfaces with high precision and repeatability.

Aggressive maneuvers with aerial robots is an area of active research with considerable effort focusing on strategies for generating sequences of controllers that stabilize the robot to a desired state. In [1, 2], Gillula et al. present an optimization-based control design methodology that generates a sequence of stabilizing controllers that drive a robot to a hover state after entering a flipping maneuver. The authors are able to provide guarantees of recovery from a flipping maneuver based on the robot model and present experimental results to validate their approach. Tedrake proposes an alternate optimization-based design methodology with similar guarantees but does so by using a guided sparse sampling of the state space and creating sequences of stabilizing controllers that drive the system to a desired state based on this sampling [3].

Impressive results are also shown using methods based on reinforcement learning or iterative adaptation of the control law. In [4], Lupashin et al. propose a control law with initial parameters for flipping a quadrotor multiple times. The control law is executed many times and corrected after each trial toward the desired performance. A similar strategy is presented in [5, 6], where the authors develop a minimal control law model and refine the model based on data collected from an expert human operator executing the aggressive maneuver. In both cases, system models are based on first principles.

In contrast to the work presented in [1]– [6], we address the challenge of designing trajectories in the full, 12-dimensional state space with an underactuated robot with four actuators. Specifically, we consider goal states parameterized by an arbitrary position, linear velocity, pitch, two angles of orientation and their derivatives. We depart from the optimization-based methods described in [1–3] because these methods do not appear to scale to 12 dimensions. Additionally, it is unclear how to incorporate differences between the first-principles model and actual model in these methods. The apprenticeship methods in [5, 6] require an expert human operator to generate data for model and control identification and therefore limit the ability of the control law to handle cases not considered *a priori* by the human operator.

We develop a system model based on first principles (Sect. 1.2) and feedback control laws for families of trajectories (Sect. 1.3). A trajectory consists of a sequence of trajectory segments and associated controllers, each of which are refined in simulation and experimentation (Sect. 1.4). Experimental evaluation of the approach shows that with limited experimental refinement (four trials of parameter adaptation) we are able to generate controllers that show repeatability and precision (Sect. 1.6).

1.2 Modeling

1.2.1 Dynamic Model

The coordinate systems and free body diagram for the quadrotor are shown in Fig. 1.1(b). 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. 1.1(b)). 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

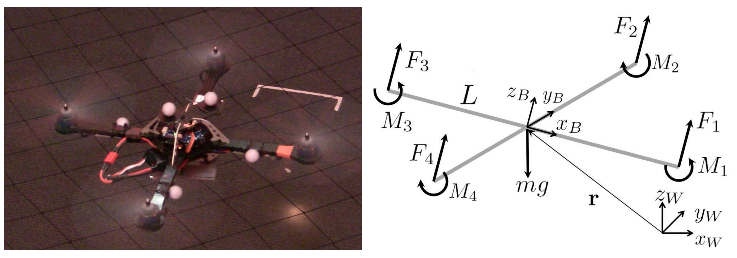
$$m\ddot{\mathbf{r}} = \begin{bmatrix} 0 \\ 0 \\ -mg \end{bmatrix} + R \begin{bmatrix} 0 \\ 0 \\ \Sigma F_i \end{bmatrix}. \tag{1.1}$$

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 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. [1] The angular acceleration determined by the Euler equations is

[1] The off-diagonal terms of I are nearly zero since $x_B - y_B - z_B$ are close to the principal axes of the quadrotor.



(a) (b)

Fig. 1.1. (a) Quadrotor used in experiments. (b) Coordinate systems and forces/moments acting on the quadrotor.

$$I\begin{bmatrix}\dot{p}\\ \dot{q}\\ \dot{r}\end{bmatrix} = \begin{bmatrix}L(F_2 - F_4)\\ L(F_3 - F_1)\\ M_1 - M_2 + M_3 - M_4\end{bmatrix} - \begin{bmatrix}p\\ q\\ r\end{bmatrix} \times I \begin{bmatrix}p\\ q\\ r\end{bmatrix}. \tag{1.2}$$

1.2.2 Motor Model

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

$$F_i = k_F \omega_i^2. \tag{1.3}$$

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

$$M_i = k_M \omega_i^2. \tag{1.4}$$

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

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

$$\dot{\omega}_i = k_m(w_i^{des} - w_i).$$

This motor gain, k_m, is found to be about $20\,\mathrm{s}^{-1}$ 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 rpm and 7800 rpm.

1.3 Control

Here we present methods used to control the robot (a) to a desired attitude; (b) to hover in place; and (c) along a three-dimensional trajectory with specified position and linear velocity. We design a sequence of these controllers to generate the desired trajectories in the next section. The controllers presented in this section are also discussed in [7].

1.3.1 Attitude Control

The goal of this controller is to reach a desired attitude with a specified angular velocity. The vector of desired rotor speeds can be written as a linear combination of four terms

$$\begin{bmatrix}\omega_1^{des}\\ \omega_2^{des}\\ \omega_3^{des}\\ \omega_4^{des}\end{bmatrix} = \begin{bmatrix}1 & 0 & -1 & 1\\ 1 & 1 & 0 & -1\\ 1 & 0 & 1 & 1\\ 1 & -1 & 0 & -1\end{bmatrix}\begin{bmatrix}\omega_h + \Delta\omega_F\\ \Delta\omega_\phi\\ \Delta\omega_\theta\\ \Delta\omega_\psi\end{bmatrix}, \tag{1.5}$$

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. We use proportional derivative control laws that take the form

$$\begin{aligned}\Delta\omega_\phi &= k_{p,\phi}(\phi^{des} - \phi) + k_{d,\phi}(p^{des} - p)\\ \Delta\omega_\theta &= k_{p,\theta}(\theta^{des} - \theta) + k_{d,\theta}(q^{des} - q)\\ \Delta\omega_\psi &= k_{p,\psi}(\psi^{des} - \psi) + k_{d,\psi}(r^{des} - r).\end{aligned} \tag{1.6}$$

Substituting (1.6) into (1.5) yields the desired rotor speeds.

1.3.2 Hover Controller

The goal of this controller is to reach a desired position and yaw angle with zero linear and angular velocities. 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, $\ddot{\mathbf{r}}_i^{des}$, are calculated from PID feedback of the position error, $e_i = (\mathbf{r}_{i,T} - \mathbf{r}_i)$, as

$$(\ddot{\mathbf{r}}_{i,T} - \ddot{\mathbf{r}}_i^{des}) + k_{d,i}(\dot{\mathbf{r}}_{i,T} - \dot{\mathbf{r}}_i) + k_{p,i}(\mathbf{r}_{i,T} - \mathbf{r}_i) + k_{i,i}\int(\mathbf{r}_{i,T} - \mathbf{r}_i) = 0,$$

where $\dot{\mathbf{r}}_{i,T} = \ddot{\mathbf{r}}_{i,T} = 0$ for hover. We linearize (1.1) to get the relationship between the desired accelerations and roll and pitch angles

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

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

$$\phi^{des} = \frac{1}{g}(\ddot{\mathbf{r}}_1^{des}\sin\psi_T - \ddot{\mathbf{r}}_2^{des}\cos\psi_T) \tag{1.8a}$$

$$\theta^{des} = \frac{1}{g}(\ddot{\mathbf{r}}_1^{des}\cos\psi_T + \ddot{\mathbf{r}}_2^{des}\sin\psi_T) \tag{1.8b}$$

$$\Delta\omega_F = \frac{m}{8k_F\omega_h}\ddot{\mathbf{r}}_3^{des}. \tag{1.8c}$$

We use two sets of position gains ($k_{p,i}$, $k_{p,i}$, and $k_{i,i}$) for the Hover Controller. The first set of gains are designed to minimize steady state error, resulting in a stiff position controller. We call this the "Stiff Hover Control". The second set of gains are smaller in order to increase the region of convergence. These gains result in a slower, more damped response and we call this the "Soft Hover Control."

1.3.3 3D Trajectory Control

The trajectory controller is used to follow 3D trajectories with modest accelerations so the linearization about the hover state is still acceptable. We use an approach similar to that described in [8] 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 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, $\ddot{\mathbf{r}}_{i,des}$, from PD feedback of the position and velocity errors:

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

Note that the $\ddot{\mathbf{r}}_{i,T}$ 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 (1.8a)–(1.8c) to compute the desired roll and pitch angles as well as $\Delta\omega_F$.

1.4 Trajectory Generation and Parameter Adaptation

1.4.1 Trajectory Description

We design a sequence of controllers to reach a goal state, G, with a specified position, x_G, velocity, v_G, yaw angle, ψ_G, and pitch angle, θ_G, with zero angular velocity and roll angle. The sequence consists of 5 phases:

- Phase 1 - hover control (stiff) to a desired position (Sect. 1.3.2);
- Phase 2 - control to desired velocity vector (Sect. 1.3.3);
- Phase 3 - control to desired pitch angle (Sect. 1.3.1);
- Phase 4 - control to zero pitch angle;
- Phase 5 - hover control (soft) to a desired position.

The yaw angle is controlled to be ψ_G during all phases. In phase 2, the robot controls along a 3D line segment at a commanded velocity towards a "launch point." Phase 3 is initiated when the quadrotor passes the plane perpendicular to the desired velocity at the launch point. In phase 3, the robot's attitude is controlled to a commanded pitch angle and a roll angle of zero. Note that during phase 3, a constant net thrust is commanded. Phase 4 and 5 are recovery phases. In phase 4, we use the attitude controller to control to a pitch and roll angle of zero. In phase 5, a soft hover controller is used to stabilize to a position in space with zero velocity.

1.4.2 Initial Parameter Selection

The pitch tracking controller used in phase 3 is tuned so that the settling time is approximately 0.4 seconds and response is close to critically damped in response to a step input between $45°$ and $120°$. For this reason, we design the system to reach state G 0.4 seconds after starting phase 3. We find the position of the launch point, x_L, and the velocity vector at the launch point, v_L, necessary to achieve the desired state G. This is accomplished via backwards integration of the equations of motion (1.1) from G to L assuming the pitch angle tracks a critical damping trajectory with a settling time 0.4 seconds, the yaw angle is ψ_G, the roll angle is zero, and the net thrust is equal to the desired thrust. During phase 2, the quadrotor starts at x_S and follows the line segment from x_S to x_L with velocity v_L. So the start position, x_S, is found by drawing a line segment of length l, typically 1 m, in the direction of $-v_L$ from x_L.

1.4.3 Parameter Adaptation

The real quadrotor does not perform exactly the same as the model. The system will not reach the exact launch point, x_L, or have the exact desired velocity, v_L, at the launch point. Additionally the quadrotor will not have an exactly zero pitch angle at the launch point and the angle performance will not be exactly critically damped with a settling time of 0.4 seconds. These deviations are caused by air drag, rotor dynamics, and actuator saturation limits. It is difficult to model these effects precisely so we iterate on experimental trials to achieve the goal state G. The initial parameter solution is run on the system once. Then we iterate n_a times on the commanded pitch angle during phase 3. We let the commanded pitch angle at iteration k be θ_C^k. We let θ_{act}^k be the actual pitch angle achieved 0.4 seconds after entering phase 3 during iteration k and update with a step size parameter, $\gamma_\theta \leq 1$, as follows

$$\theta_C^{k+1} = \theta_C^k + \gamma_\theta(\theta_G - \theta_{act}^k) \tag{1.9}$$

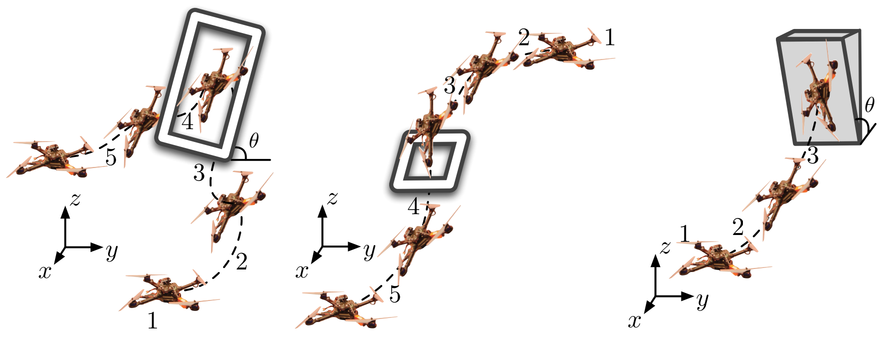
The pitch angle achieved during phase 3 is not strongly affected by the velocity commanded during phase 2, so we can iterate on the commanded velocity without significantly affecting the pitch angle. We use the same strategy as for pitch angle and iterate n_v more times on velocity:

$$v_C^{k+1} = v_C^k + \gamma_v(v_G - v_{act}^k) \tag{1.10}$$

Finally, we run the final parameters for n_x trials and let $\bar{x}_{act}$ be the average position achieved 0.4 seconds after entering phase 3 during the trials. The entire trajectory is then simply shifted by the difference between the desired position, x_G, and the actual position, $\bar{x}_{act}$, as follows

$$\begin{aligned} x_L &= x_L + (x_G - \bar{x}_{act}) \\ x_S &= x_S + (x_G - \bar{x}_{act}) \end{aligned}$$

Note that the gains for all the controllers are designed ahead of time. During parameter adaptation only the commanded pitch angle and the three components of the commanded velocity are modified.



(a) Vertical opening (b) Horizontal opening (c) Perching

Fig. 1.2. The four experimental scenarios considered in this work. Flying downward and upward through a horizontal opening are both shown in Fig. 1.2(b), where the stages of the robot's progression are reversed for the latter. Note that in Figs. 1.2(a) and 1.2(c), θ denotes the varied window and perching orientation.

1.5 Experiment Design and Implementation Details

In this work we present a systematic approach for designing trajectories and associated controllers that permit aggressive maneuvers with quadrotor robots. We consider four experimental scenarios:

- flying through a vertical opening at varying angles;
- flying downward through a horizontal opening;
- flying upward through a horizontal opening;
- perching on a target at varying angles.

Note that adhesion during perching is achieved by placing Velcro on the underside of the quadrotor and on a target location. Graphics depicting the four scenarios are shown in Fig. 1.2. The first and last scenarios include cases at various angles. For each of the cases, the quadrotor executed 15 trials of the trajectory after completing 2 iterations of (1.9) and 4 iterations of (1.10). We report the results of these experiments in Sect. 1.6.

The hardware, software, and implementation details of the experiments follows. The pose of the quadrotor is observed using a VICON motion capture system at 225 Hz [9]. The position is numerically differentiated to compute the linear velocity of the robot. These values are available to MATLAB via ROS [10] and a ROS-MATLAB bridge [11]. All commands are computed in MATLAB using the latest state estimate at the rate of the VICON. The commands in MATLAB are bridged to ROS and the most recent command is sent to the robot via ZIGBEE at a fixed rate of 100 Hz. This fixed rate is due to the limited bandwidth of ZIGBEE (57.6 kbps). Commands sent to the robot consist of the gains and desired attitude, desired angular velocities, and thrust values described in Sect. 1.3.1.

The robot (Fig. 1.1(a)) is sold commercially [12]. The quadrotor follows a standard four-propeller design and is equipped with two embedded processors running at

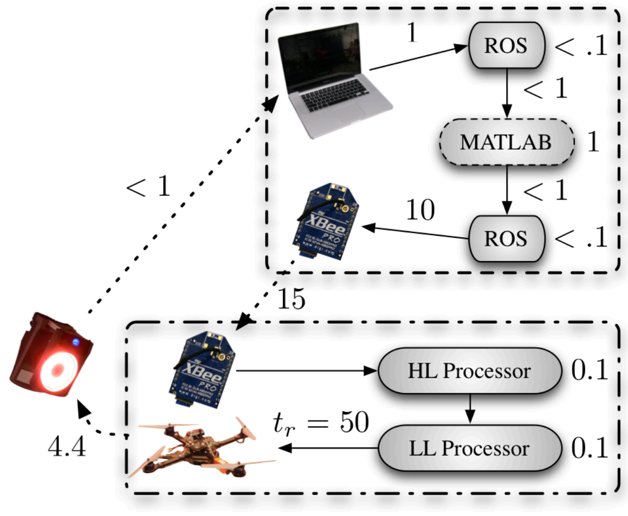


Fig. 1.3. Latencies (ms) in experimental system. The motion capture system operates at 225 Hz. Quadrotor pose and velocity data is received and bridged to Matlab. Commands are computed in Matlab and the latest command is sent to the robot via Zigbee at 100 Hz (throttled due to bandwidth limitations). Commands are received on the robot; the high-level (HL) embedded micro-processor computes direct motor commands using (1.5) and (1.6), then sends the motor commands to the lower-level (LL) processor. Our custom firmware runs on the HL processor and the proprietary firmware runs on the LL processor. The motor response time is denoted as t_r. Negligible times (< 0.01 ms) are not noted. The worst case response of the system from observation to motors rotating based on the observation is approximately 85 ms with t_r as the dominant limiting factor.

1 kHz (denoted as high-level (HL) and low-level (LL)), an IMU (running at 300 Hz), and other sensors not required for this work. The HL processor runs our custom firmware, receives commands via ZIGBEE, and sends motor commands to the LL processor to execute those commands. The LL processor provides attitude estimates to the HL processor. A comparison between attitude estimates and VICON shows very similar results. Latencies and data flow are shown in Fig. 1.3.

1.6 Results

Nine cases of the four scenarios shown in Fig. 1.2 were tested. All of the cases use a desired yaw angle, ψ_G, of 90°. The details of these cases are shown in Table 1.1. In all of the vertical window cases (1-4) the desired velocity is 2 m/s through the window and zero in the other directions. This speed is large enough for the quadrotor to coast through the window at the desired angle. For descending though the horizontal window (case 5), the desired downward speed is 1.5 m/s and zero in the other directions. This speed has to be small enough to give the quadrotor time to recover after passing through the window. Ascending through the horizontal window (case 6)

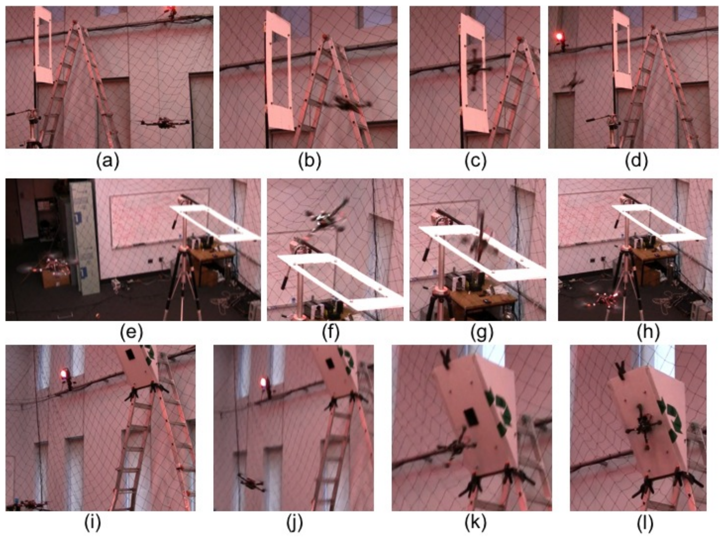


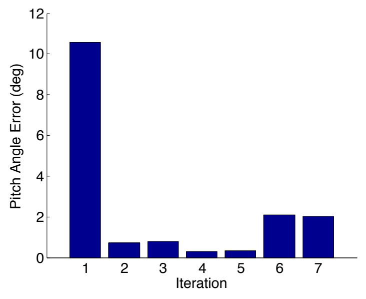
Fig. 1.4. Images from representative experimental trials. Figures 1.4(a-d) present a trial of the quadrotor passing through a vertical window at 90° (case 4). Figures 1.4(e-h) show the quadrotor descending through a horizontal window (case 5). Figures 1.4(i-l) show the quadrotor perching on a 120° surface (case 9). Videos of the experiments are available at http://tinyurl.com/quadrotorcontrol.

Case	Description	v_G (x,y,z) (m/s)
1	Vertical Window at 45°	(2, 0, 0)
2	Vertical Window at 60°	(2, 0, 0)
3	Vertical Window at 75°	(2, 0, 0)
4	Vertical Window at 90°	(2, 0, 0)
5	Down Through Horizontal Window at 90°	(0, 0, -1.5)
6	Up Through Horizontal Window at 90°	(0, 0.4, 2.2)
7	Perch at 60°	(0, 0.8$\cos(30)$, -0.8$\sin(30)$)
8	Perch at 90°	(0, 0.8, 0)
9	Perch at 120°	(0, 0.8$\cos(30)$, 0.8$\sin(30)$)

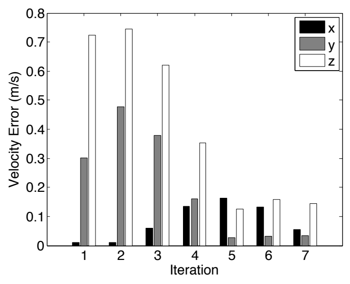
Table 1.1. Descriptions of tested cases.

is the most difficult case because the quadrotor must achieve enough vertical speed to coast upward through the window. For each of the perching cases (7-9), the desired velocity was set to be 0.8 m/s normal to the perching surface. This speed is large enough to guarantee adhesion given proper alignment of the quadrotor to the perching surface. Representative images from cases 4, 5, and 9 are shown in Fig. 1.4.

The performance of the iteration scheme is shown for a representative case (case 8) in Fig. 1.5. After the first iteration the initial angle error drops from $10°$ to less than $2°$ for the rest of the iterations. The velocity adaptation begins after iteration 3. The velocity error improves significantly in iteration 4 and continues to stay low for

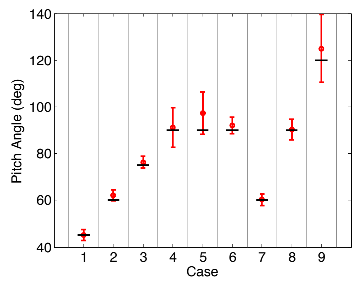


(a)

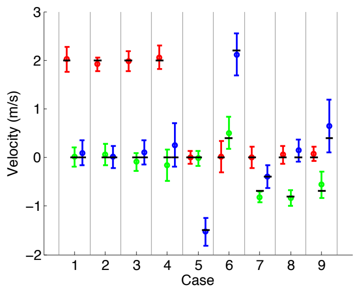


(b)

Fig. 1.5. Pitch angle and velocity improvement after iterating when perching at $90°$ (case 8).



(a)



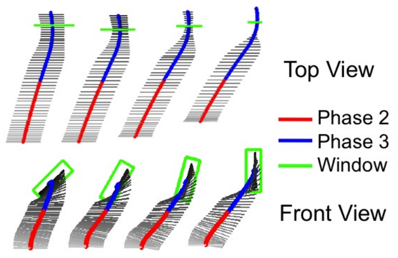
(b)

Fig. 1.6. Mean pitch angles (a) and velocities (b) for 15 trials of each case. The error bars represent three standards deviations and the black bars are the desired pitch angles and velocities. In (b) the x, y, and z velocities are shown in red, green, and blue, respectively.

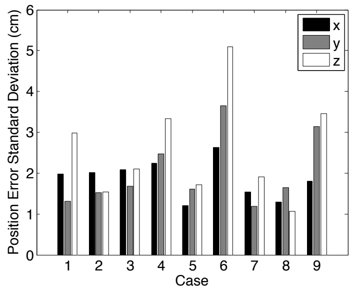
the remainder of the iterations. After performing the iteration scheme for each of the cases, 15 trials with the same parameters are run for each case. A summary of the data collected from the 15 trials for all of the cases is shown in Figs. 1.6 and 1.7(b). For a representative case (case 3) the standard deviation on the achieved angle is $0.9°$ and the standard deviations on the velocity and position are around 8 cm/s and 2 cm, respectively, for all axes. To illustrate the trajectory for this case and the other vertical window cases (1-4), a single trial for each is shown in Fig. 1.7(a).

1.7 Conclusion and Future Work

In this paper we study the problem of designing dynamically feasible trajectories and controllers that drive a quadrotor to a desired state in state space. We focus on the development of a family of trajectories defined as a sequence of segments, each with a controller parameterized by a goal state. Each controller is developed from the dynamic model of the robot and then iteratively refined through successive experi-



(a)



(b)

Fig. 1.7. (a) Experimental data from a single trial for cases 1-4, the vertical window maneuver at 45°, 60°, 75°, and 90° (left to right). The gray lines represent the orientation of the quadrotor. (b) Standard deviations on goal positions for 15 trials for each case.

mental trials to account for errors in the dynamic model and noise in the actuators and sensors. Four scenarios are tested experimentally as considered by nine case studies with fifteen trials of each case study. The scenarios include flying through narrow, vertical and horizontal openings and perching on an inverted surface. We show that our approach results in repeatable and precise control along trajectories that demand velocities and accelerations that approach the limits of the vehicle's capabilities.

Much of our approach is moving toward planning for dynamically feasible trajectories that require more sequences of controllers than the number shown here (at most five in this work). We believe that by representing families of trajectories by specialized controllers the high-dimensionality of the planning problem may be reduced and so we are actively pursuing this area of research. We are also pursuing the extension of our approach to external disturbances such as a gust of wind. Finally, we are interested in considering more advanced methods for optimizing and adapting the trajectory parameterizations to deal with differences between the analytic model and reality.

References

1. J. H. Gillula, H. Huang, M. P. Vitus, and C. J. Tomlin, "Design and analysis of hybrid systems, with applications to robotic aerial vehicles," in *Proc. of the Int. Symposium of Robotics Research*, Lucerne, Switzerland, Sept. 2009.
2. ——, "Design of guaranteed safe maneuvers using reachable sets: Autonomous quadrotor aerobatics in theory and practice," in *Proc. of the IEEE Int. Conf. on Robotics and Automation*, Anchorage, AK, May 2010, pp. 1649–1654.
3. R. Tedrake, "LQR-Trees: Feedback motion planning on sparse randomized trees," in *Proc. of Robotics: Science and Systems*, Seattle, WA, June 2009.
4. S. Lupashin, A. Schollig, M. Sherback, and R. D'Andrea, "A simple learning strategy for high-speed quadrocopter multi-flips," in *Proc. of the IEEE Int. Conf. on Robotics and Automation*, Anchorage, AK, May 2010, pp. 1642–1648.

5. J. Tang, A. Singh, N. Goehausen, and P. Abbeel, “Parameterized maneuver learning for autonomous helicopter flight,” in *Proc. of the IEEE Int. Conf. on Robotics and Automation*, Anchorage, AK, May 2010, pp. 1142–1148.
6. P. Abbeel, “Apprenticeship learning and reinforcement learning with application to robotic control,” Ph.D. dissertation, Stanford University, Stanford, CA, Aug. 2008.
7. N. Michael, D. Mellinger, Q. Lindsey, and V. Kumar, “The GRASP multiple micro UAV testbed,” *IEEE Robotics and Automation Magazine*, vol. 17, no. 3, pp. 56 –65, Sept. 2010.
8. 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.
9. “Vicon Motion Systems, Inc.” http://www.vicon.com.
10. “Robot Operating System (ROS),” http://www.ros.org.
11. “ROS-Matlab Bridge,” http://github.com/nmichael/ipc-bridge.
12. “Ascending Technologies, GmbH,” http://www.asctec.de.