Cooperative Control and Modeling for Narrow Passage Traversal with an Ornithopter MAV and Lightweight Ground Station

Ryan C. Julian
Department of EECS
Univ. of California, Berkeley
Berkeley, CA 94720
ryanjulian@berkeley.edu

Cameron J. Rose
Department of EECS
Univ. of California, Berkeley
Berkeley, CA 94720
c_rose@eecs.berkeley.edu

Humphrey Hu
Robotics Institute
Carnegie Mellon University
Pittsburgh, PA 15213
humhu@cmu.edu

Ronald S. Fearing
Department of EECS
Univ. of California, Berkeley
Berkeley, CA 94720
ronf@eecs.berkeley.edu

ABSTRACT

The power, size, and weight constraints of micro air vehicles (MAVs) limit their on-board sensing and computational resources. Ground vehicles have less mobility than MAVs, but relaxed size constraints, and typically more computing power. These specializations present many opportunities for robot-robot cooperation. In this work, we demonstrate cooperative target-seeking between a 13 gram ornithopter MAV and a lightweight ground station using computer vision. We develop models for the ornithopter, ground station, and cooperative system dynamics. We determine model parameters of the real systems through experimental system identification. Finally, we verify those models using experiments on narrow passage traversal, and arrive at a cooperative system model which accurately predicts the backwards-reachable region for successfully negotiating ornithopter flight through narrow passages.

We also introduce a new ornithopter MAV, the 13 gram H^{2}Bird. It features clap and fling wings, improves upon previous designs by utilizing a carbon fiber airframe, tail rotor, and elevator, and carries a 2.8 gram payload. We augment the ornithopter's built-in gyroscope-based control with a lightweight ground station, which has power and weight requirements appropriate for deployment on ground vehicles with 10 gram payloads. The ground station provides heading estimates to the ornithopter by running a real-time motion tracking algorithm over a live video stream.

Categories and Subject Descriptors

I.2.9 [**Robotics**]: Autonomous Vehicles; I.2.9 [**Robotics**]: Distributed Artificial Intelligence—*Multiagent Systems*; I.2.9 [**Robotics**]: Vision and Scene Understanding—*Motion*

Appears in: *Proceedings of the 12th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2013), Ito, Jonker, Gini, and Shehory (eds.), May, 6–10, 2013, Saint Paul, Minnesota, USA.*

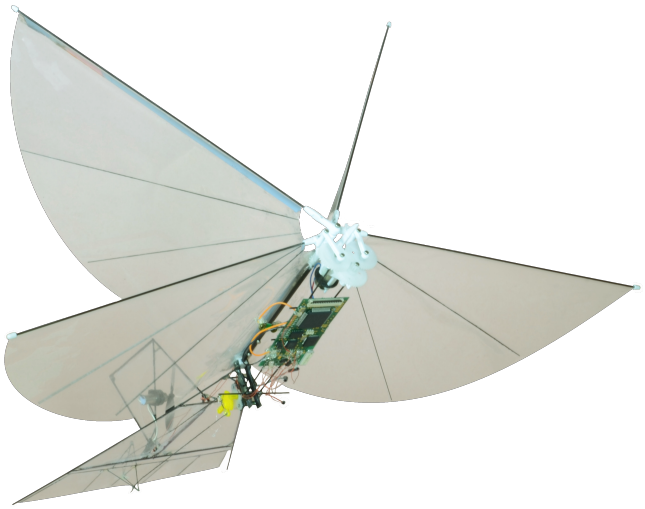

Figure 1: H^{2}Bird ornithopter MAV

General Terms

Algorithms, Performance, Design, Experimentation

Keywords

AAMAS proceedings, multiagent systems, cooperative control, particle filter, micro air vehicle, ground station, visual servoing, pose estimation, flapping wing, ornithopter, biomimetic

1. INTRODUCTION

Power, size, and weight constraints on 10 gram scale micro air vehicles (MAVs) significantly limit their sensing and computational capabilities. MAVs can execute only low-complexity control and state estimation algorithms in real-time. The constraints on flapping-wing MAVs are even more restrictive. Although they demonstrate safety and noise profiles superior to those of rotorcraft MAVs, this comes at the expense of reduced payload.

The non-linear dynamics of flapping-wing flight compound these constraints [4][12]. Much past research on flapping-wing MAVs focuses on overcoming the difficulties of controlling and modeling flapping-wing vehicles in the face of this non-linearity [9]. Common approaches to simplify these dynamic models include averaging over the wing-beat period [4][25], and using linear, low-dimensional models to predict flight behavior [26]. These advances led to the development of high-performance platforms such as the sub-15 gram DelFly, described by deCroon et al. [5].

Many control laws, such as periodic forcing and other control methods described by [7][15][19], require powerful onboard computing. In this work, we develop a simpler, less computationally-intensive method of control for negotiating narrow passages, by exploiting cooperation between specialized robotic agents.

In contrast to flight control, there is considerably less previous work on guidance and navigation of flapping-wing MAVs. Baek demonstrated altitude control with an external camera [2]. de Croon et al. demonstrated obstacle avoidance with an onboard camera and offboard processing [6]. Baek et al. provide one of the few examples of completely autonomous target seeking for vehicles at the sub-20 gram scale [3].

There also exists a body of work on the guidance and control of fixed- and rotary-wing MAV platforms, but many of these methods make extensive use of GPS or motion capture systems [14][16]. Indoors, using GPS to estimate the pose of an MAV is unreliable, and motion capture requires extensive and stationary modification of the environment.

The literature on cooperative control of MAVs is similarly limited to larger vehicles or analytical explorations. Hyams et al., Jung et al. and Nebot et al. all demonstrated cooperative visual servoing for heterogeneous UGVs [11][13][22]. Luo, et al. and Mehta, et al. analyze UAV-UGV cooperation from a modeling perspective [20][21]. Rudol et al. demonstrated a similar cooperative visual servoing system with UGVs and MAVs [24], but the vehicles used were orders of magnitude larger than the lightweight systems we target, and so were able to take advantage of high-performance processors and specialized cameras. Stirling et al. describe a method for cooperation amongst quadrotor MAVs to navigate an indoor environment [27]. Stationary MAVs provide external sensing for navigation through the environment, and the network can be extended by using the exploring MAVs for sensing in undiscovered areas. Their implementation relies solely on local sensing, although our size range requires that the actual environment sensing be external to the MAV. Additionally, they do not consider time constraints in their planning, whereas our MAV has a minimum forward speed and turning radius. The small size of lightweight air and ground vehicles necessitates specialization, but it also pays dividends in agility. A group of specialized heterogeneous, lightweight robots can navigate more challenging terrains and interact with their environments in ways that larger, monolithic vehicles cannot. Despite this advantage, in order to collectively equal the sensing and computational capabilities of larger robots, small robots will have to interact so that multiple robots can benefit from the specialized capabilities of cooperating vehicles. In this paper we consider both experimentally, and in simulation, the performance of the externally-directed ornithopter to demonstrate how millirobotic system can benefit from cooperation.

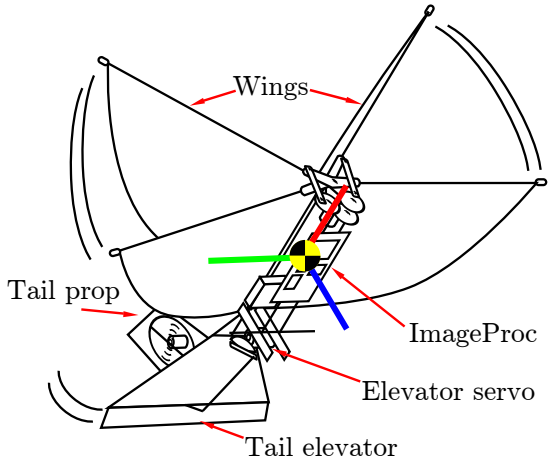

Figure 2: H^{2}Bird ornithopter with attitude control axes and labeled control surfaces.

2. ROBOTIC AND VISION PLATFORMS

We develop a new flapping-winged MAV, the H^{2}Bird, and ground station. We use these systems to cooperatively navigate through a narrow passage, by sensing the position of the MAV and the goal using the ground station.

2.1 H^{2}Bird Ornithopter

To explore these cooperative control concepts, we designed a new flapping-wing MAV, known as the H^{2}Bird (Figure 1). Built around the Silverlit i-Bird RC flyer power train and clap-fling wings[1], the H^{2}Bird has a wingspan of 26.5 cm and a flight weight of 13 grams. A tail propeller and servo-controlled tail elevator provide control in the yaw and pitch axes (Figure 2). The on-board ImageProc 2.4[2] controller includes a 40 MIPS microprocessor, 6 DOF IMU, IEEE 802.15.4 radio, VGA camera, and motor drivers, all powered by a 90 mAh lithium polymer battery. In routine flight, the H^{2}Bird averages 1.2 m/s ground speed, and operates for approximately 10 minutes.

2.2 Ground Station Computer Vision Platform

Cooperative behavior is a promising strategy for autonomous navigation in unexplored and unstructured environments. To operate in these environments, a cooperative system must be fully mobile, not tied to a static ground station or motion capture laboratory. To demonstrate the feasibility of this approach, we designed our ground station to conform to the power and weight constraints of highly mobile millirobotic ground vehicles, such as OctoRoACH, which can traverse rough terrains with speed and agility [23].

Rather than power-intensive PC processors, we used the ARM-based BeagleBoard single-board computer as a reference computation platform. The BeagleBoard consumes approximately one watt while running our computer vision and control algorithms. It uses the same processor as popular sub-10 gram processor modules–such as the Gumstix Overo and LogicPD Torpedo–which are light enough for deployment on lightweight ground vehicles. We pair the Bea-

[1]Silverlit Toys Manufactory Ltd.: i-Bird RC Flyer http://www.silverlit-flyingclub.com/wingsmaster/

[2]ImageProc 2.4: https://github.com/biomimetics/imageproc_pcb

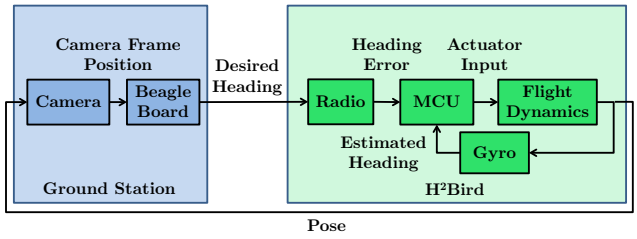

Figure 3: Overview of the cooperative control system.

gleBoard with an off-the-shelf consumer USB web camera, for similar image quality and resolution to the miniature camera modules available at millirobot scale. The BeagleBoard communicates with the H^{2}Bird via a USB radio module which implements the IEEE 802.15.4 wireless standard.

Our ground station software is based on the Ubuntu Linux[3] distribution modified with a custom version of the Linux kernel. We use the Python[4] programming language for control, robot communication, and telemetry, and the OpenCV[5] computer vision libraries for image capture and processing. All software and hardware we developed for this system is open source and freely available under a BSD license[6].

3. SYSTEM CONCEPT AND ALGORITHMS

Our system combines both the sensing of the H^{2}Bird and the execution of the desired headings to steer the robot towards the goal. The sensing is performed on the ground station using frame differencing and a particle filter. The H^{2}Bird steers to the heading directed by the ground station using a quaternion-based proportional-integral-derivative controller.

3.1 Ornithopter Attitude Control

We implement attitude estimation and control of the H^{2}Bird on-board to achieve high sample-rate attitude control. We estimate the vehicle pose by integrating the IMU, and use a proportional-integral-derivative (PID) controller to feed these estimates back to the attitude control flight surfaces (Figures 3 and 4).

The high pitch angles at which flapping-wing MAVs regularly operate distinguish them from other MAVs. Baek demonstrated control on a similar vehicle using PID and an Euler angle parameterization of orientation [3]. However, Euler angle parameterizations of vehicle orientation suffer from singularities around high pitch angles. So we instead represent the vehicle orientation with quaternions. The quaternion representation allows us to represent relative body and world coordinate angular displacements compactly with quaternion multiplication [10]. Our implementation of quaternion-based control is similar to Knoebe's [17]: Given a reference pose represented in quaternion form q_r, we define the error as the rotation q_e required to reach the reference pose from the estimated pose q:

$$q_e = q' \cdot q_r \tag{1}$$
$$= q_{e,w} + q_{e,x}\hat{i} + q_{e,y}\hat{j} + q_{e,z}\hat{k}$$

We then convert this error quaternion to an angle axis representation and use it as input to the PID controller (Equations 2 and 3). We perform these calculations on-board, with trigonometric lookup tables for real-time performance.

$$Q_e = \alpha \cdot q_e / \sin(\alpha/2) \tag{2}$$
$$= \alpha_x\hat{i} + \alpha_y\hat{j} + \alpha_z\hat{k}$$

$$\alpha = 2\arccos(q_{e,w}) \tag{3}$$

3.2 Ground Station Pose Estimation

The ground station performs simple two-dimensional pose estimation on H^{2}Bird using a particle filter-based motion tracking algorithm [28]. This pose estimate is the feedback input to the cooperative visual servoing feedback loop, which is represented as the outermost feedback paths on the block diagrams in Figures 3 and 4. We limit our pose estimation approach to monocular two-dimensional object tracking in pixel space. This allows us to perform real-time video tracking using the modest computational resources available on the ground station, and similarly, millirobotic ground vehicles.

We initialize the particles uniformly across the frame before tracking begins. To improve numerical stability, we normalize the weights of all particles on each iteration, and only resample the particle population when the mean particle weight falls below a pre-determined threshold.

3.2.1 Motion Model

Our motion model (Equation 4) is a simple Gaussian transition centered around each particle's current position, augmented with an ϵ-random uniform sampling strategy [8]. The sampling law is as follows:

$$x_t^{[i]} \sim \begin{cases} \mathcal{N}(x_{t-1}^{[i]}, \Sigma), & \text{with probability } 1-\epsilon \\ \mathcal{U}(0, b), & \text{with probability } \epsilon \end{cases} \tag{4}$$

In Equation 4, $x_t^{[i]}$ is the position of the i^{th} particle in 640x480 pixel space at time t, Σ is a chosen covariance matrix, and b is a vector of the bounds of the pixel space. The sampling method requires that, with probability ϵ, the particle filter will choose a pixel from a uniform distribution of all of the pixels in the pixel space for the location of the i^{th} particle. The ϵ-random uniform transitions increase tracking robustness, by forcing the filter to always sample broadly across the pixel space. This helps prevent overconfidence and collapse, and decreases reacquisition delay.

In the experiments, we used a diagonal covariance matrix with a variance of 256, and an ϵ of 0.3. We find the Gaussian transition model is sufficient for tracking motion in which frame-to-frame position changes are relatively small.

3.2.2 Emission Model

The emission model weights the particles' likelihood of containing a pixel location that is part of the H^{2}Bird [8]. The particles with higher weights have a greater probability of being resampled. Our emission model implements a sampling approach to motion tracking via naïve background

[3]Ubuntu: Canonical Ltd., http://www.ubuntu.com/
[4]Python: Python Software Foundation, http://www.python.org/
[5]OpenCV: Willow Garage, http://opencv.willowgarage.com/wiki/
[6]Biomimetic Millisystems Laboratory on GitHub, https://github.com/biomimetics

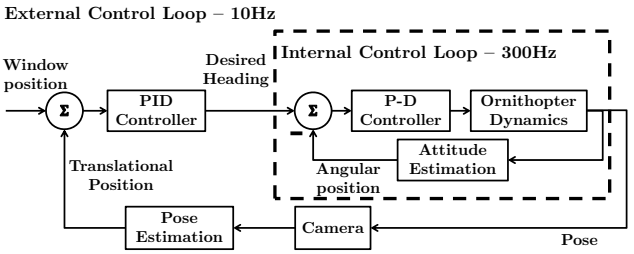

Figure 4: Overall block diagram of the cooperative control system.

subtraction, e.g. frame differencing [1]:

$$w_t^{[i]} = 1 + \|f_0(x_t^{[i]}) - f_t(x_t^{[i]})\|^2 \quad (5)$$

On each iteration, the system provides the emission model with f_0, the frame representing the background, and f_t, the most recent frame captured. The model assigns a weight $w_t^{[i]}$ to each particle $x_t^{[i]}$, proportional to the color distance in RGB space between the pixels of f_0 and f_t at the location $x_t^{[i]}$. Hence, particles are more likely to be resampled in regions of the image which are dissimilar in color from the background.

We assumed that neither the bird, nor any other moving object, were visible in the frame on system start. The first frame was captured by the system, and used as the background frame f_0 for every iteration of the particle filter.

Naïve background subtraction is brittle to changes in a scene over time [1]. However, we find it very efficient and sufficiently reliable for the short time spans–less than five seconds–necessary for H^{2}Bird to successfully negotiate narrow openings. Calculating f_0 using a more sophisticated background extraction would retain the underlying motion tracking features of the filter, while providing the algorithm with a more robust way of differentiating subject motion from the background.

3.3 Cooperative Visual Servoing

The ground station guides the H^{2}Bird by setting the reference attitude of its on-board attitude controller (Figures 3 and 4). Using the estimate of 2D vehicle pose from the pose estimation algorithm, the ground station calculates an altitude error and horizontal error between the H^{2}Bird and the goal. The ground station then uses a PID control law to calculate a new reference pitch and yaw, which it sends to the H^{2}Bird over radio. We apply a fixed-lag window to the PID controller integral term to prevent wind-up while we launch the ornithopter. We tuned the proportional and derivative terms of this outer controller to dampen the overshoot and drift caused by the cooperative system latency.

4. SYSTEM MODEL

We developed models using system identification concepts to simulate the behaviors of the H^{2}Bird and the ground station. We use these models to determine the set of initial conditions that are most likely to result in successful window traversal.

4.1 Ornithopter Attitude Control

We divide our model of the vehicle dynamics into two independent models for simplicity. We develop one model for yaw motion, and another for translational motion.

We approximate the yaw dynamics of the ornithopter with a one-dimensional model to limit complexity, and because it matches our physical implementation. The H^{2}Bird uses an on-board PID controller to follow the heading sent by the ground station, and the yaw dynamics are largely decoupled from the rest of the fight dynamics. Thus, the yaw response θ is modeled as:

$$\theta(s) = \frac{K}{1 + Rs} u(s) \quad (6)$$

with constants K and R from system identification, reference heading angle $u(s)$, and output $\theta(s)$, the physical heading angle of the H^{2}Bird. To model the translational position of the ornithopter, we use a form of Dubin's car model [18]:

$$\begin{aligned} \dot{x} &= u_s sin(\theta) \\ \dot{y} &= u_s cos(\theta) \\ \dot{\theta} &= u \\ u \in U &= [-\tan\phi_{max}, \tan\phi_{max}] \end{aligned} \quad (7)$$

In the car model, ϕ_{max} is the maximum steering angle and u_s is the forward speed. Here, the angular position θ is an input rather than a state. The overall motion of the system can be modeled as shown in Equation 7, where θ is the time-domain output of Equation 6, $\theta(t)$.

4.2 Ground Station Pose Estimation

Our model of the ground station includes models for both the camera and the pose estimation computation. Referring to Figure 4, the camera model experiences a delay in series with the path between the "Camera" and "Ornithopter Dynamics" blocks. The pose estimation algorithm experiences latency in the pose estimation block when updating the particle filter, and during the transmission of the desired heading from the external PID controller to the internal PID controller on the robot, via radio.

In addition to the latencies in the system, we model the viewing angle of the camera as a pixel map from one edge of a 63° viewing triangle to the other. The camera has an image width of 640 pixels. We calculate the desired heading angle in degrees, for a given distance from the camera:

$$\text{Desired heading} = \frac{640\,\text{px}}{2d\tan\frac{63^\circ}{2}} \times \frac{63^\circ}{640\,\text{px}} \times x \quad (8)$$

where d is the distance between the ornithopter and the camera, and x is the horizontal position of the robot relative to the center of the window in meters. Note that the pose estimation algorithm does not determine the depth of the robot, which can be problematic for control algorithms. For example, if the ornithopter remains at a constant horizontal distance from the window, but moves closer to the camera, the ground station's directed heading steadily increases, even though the robot has not actually moved farther from the window. These problems are most severe in areas close to the camera.

5. EXPERIMENTS AND RESULTS

To bound the performance of the cooperative control system, we determine the feasible set of initial conditions that

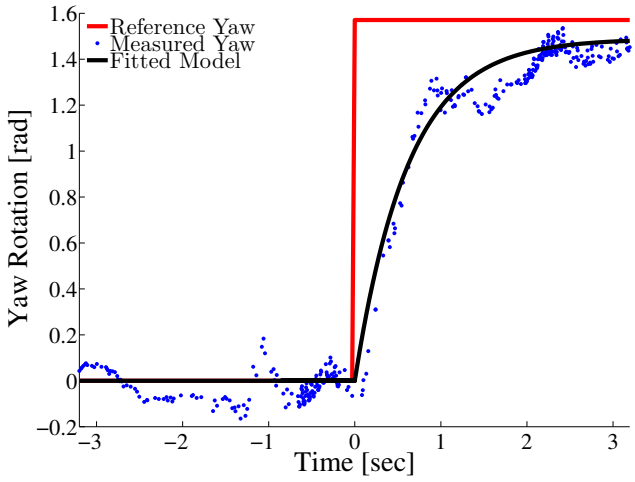

Figure 5: H^{2}Bird step response

result in successful narrow passage traversal, using both analytical and empirical methods. We then conduct experiments to validate this model.

5.1 System Identification

We determined the turning speed and turning radius of the H^{2}Bird experimentally, by measuring the response of the system to a step input of a clockwise 90° turn. We measured the response of the system by recording the attitude estimates calculated by the ornithopter using its onboard gyroscope. We then used MATLAB[7] to fit a simple low-order model to the yaw step response, as discussed in Section 4.1:

$$\theta(s) = \frac{0.95}{1 + 0.62s} u(s) \tag{9}$$

We calculated the step response rise time to be 1.4 seconds (Figure 5). The H^{2}Bird flies at an average forward velocity of 1.2 m/s, thus we estimate the minimum turning radius of the robot to be 1.07 meters:

$$\begin{aligned} \text{Minimum radius} &= \frac{360° \times \text{Rise time} \times \text{Flight speed}}{\text{Final angle} \times 2\pi} \\ &= \frac{360° \times 1.4s \times 1.2\frac{m}{s}}{90° \times 2 * \pi} = 1.07m \end{aligned} \tag{10}$$

In the experiments, we choose a flight speed lower than the maximum speed of the ornithopter, to facilitate more robust tracking.

To measure the capture latency of the camera, we wrote a simple test application which draws a large colored rectangle on the ground station's screen, then uses the camera to detect the changes in the rectangle's color. By noting the value of the system timer just before the rectangle changes color and just after the camera detects the change, the application records the estimate of the video capture pipeline latency. We averaged the result over several hundred rapidly-executed trials to determine the capture latency we used to model the system.

[7]The MathWorks, Inc. Matlab: http://www.mathworks.com/products/matlab/

To compute the particle filter and controller latency, we again used system timers to determine the average time between the start and end of computation. We measured a camera capture latency of 40 ms, pose estimation computation latency of 12.5 ms, and radio transmission latency of 25 ms.

5.2 Model Verification by Simulation

We determine the backwards reachable set of initial states for successful narrow passage traversal both geometrically and using Monte Carlo simulation. We also simulate the motion of the ornithopter beginning from various position and angular heading initial conditions. We use our model, as determined through system identification and described in Section 5.1, using the cooperative control method in Figure 4. The results are presented in Figures 6 and 7.

In Figure 6, the blue lines represent the camera viewing triangle, the black line represents the window, and the black dot represents the camera. For the initial conditions, we used a uniform grid of 5 cm increments within the camera viewing region, and an initial angular heading perpendicular to the window plane. A success, or feasible initial condition, consists of a path that intersects the window at some point and does not leave the camera viewing triangle, and is indicated in green. The geometrically infeasible region is computed using the minimum turning radius and the geometry of the camera's field-of-view, and is indicated in blue. The region in red is geometrically feasible, but is infeasible as a result of the cooperative control simulation dynamics.

We conducted a Monte Carlo simulation to determine the probability of successful window navigation for a given point within the testing area. Using this information we can make assumptions about ideal starting positions for cooperative control. For each point in a 10 cm grid within the camera viewing triangle, we simulated 40 trials using an initial heading randomly sampled from a uniform distribution between -90° and 90°, with the 0° heading perpendicular to the window. Figure 7 depicts the results of the Monte Carlo simulation.

As shown in the figure, there is a region in the middle where we had success in all 40 of the trials, which is a subset of the feasible region presented in Figure 6. Along the edges of the camera viewing triangle, however, the probability of success decreases.

5.3 Experiments

We tested our cooperative visual servoing system with a simple target seeking experiment: the H^{2}Bird must fly through a specified obstacle, in this case, a wooden frame we used to simulate a window. The ground station provides remote guidance to H^{2}Bird to allow it to navigate through the window. A motion capture system records ground truth data. Our setup is shown in Figure 8.

Each trial included the following steps (Figure 9):

(1,2): We identify the window to the ground station by dragging a bounding box over it using a mouse.

(3): We release the H^{2}Bird by hand, in view of the camera at the desired starting grid point.

(4-6): The H^{2}Bird attempts to fly through the window, with remote guidance from the ground station.

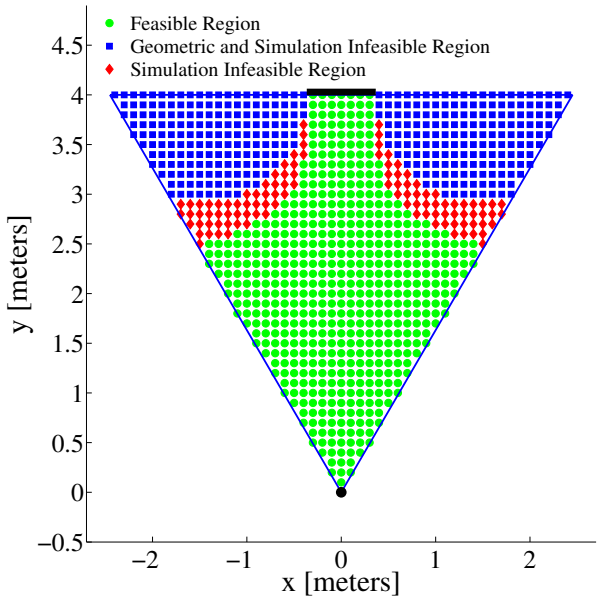

Figure 6: Plot of backwards reachable set for successful window traversal.

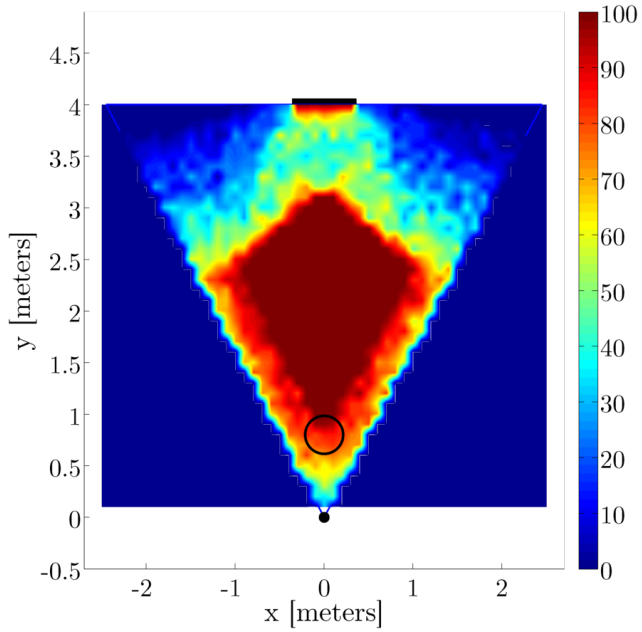

Figure 7: Probability of success, resulting from a Monte Carlo simulation of the reachable set model, for all starting locations.

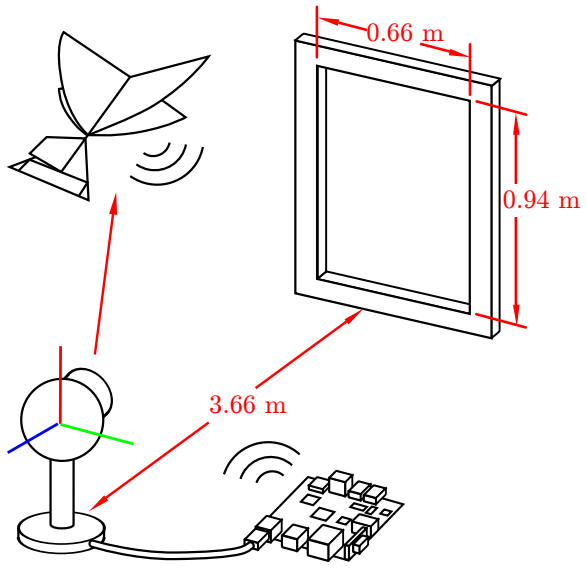

Figure 8: Conceptual sketch of experimental setup.

Every 5 trials, we replaced the ornithopter's battery with a fully charged one.

We collected 80 trials over a variety of starting positions. Our goal for these experiments was to verify the model described in Section 4, so we conducted 60 of the 80 trials in a grid along the edges of the camera view plane, and in 5 cm increments from the window plane, in an effort to determine the success rate on the edges of the testing space. We conducted 20 additional trials directly in front of the camera to determine the success rate near the center of viewing plane.

The H^{2}Bird achieved a success rate of 80% for initial conditions in the middle of the testing space and in front of the camera. As the H^{2}Bird moved towards the edges of the feasible region in Figure 11, the success rate diminished to 50%.

The lower success rate on the edges of the camera frame can be explained by several factors. On the edges of the viewing triangle, the yaw control inputs can be large in magnitude. Since the control algorithm has no notion of depth, it applies the same control input whether the ornithopter is close to the camera, where small changes in heading are adequate, or far away from the camera, where large changes in heading are necessary. Due to this phenomenon, the controller often over or under-compensates in certain regions, depending upon the PID tuning and the distance from the camera. Tracking noise and variations in the H^{2}Bird hand launch can cause the system to respond differently for similar initial conditions, or to fail completely. In addition, the H^{2}Bird performance is affected by battery power variation.

The results of our experimental trials verify the results of the Monte Carlo simulation of our model. The trials that we conducted in Figure 10 correspond to the circled region in Figure 7. The 80% success rate that we measured in the experiments corroborates our calculated probability of success of approximately 80%, demonstrated by the Monte Carlo simulation. Additionally, the diminished success rate of 50% near the edges of the viewing triangle that we determined through experimentation supports the reduced probability of success calculated by the simulation.

Our experimental data validates Figure 6, as there are no successes within the geometrically infeasible region. There are few successes near the boundary of the infeasible region determined in simulation. While there are successes along the boundary of the camera viewing triangle, there are also many failures.

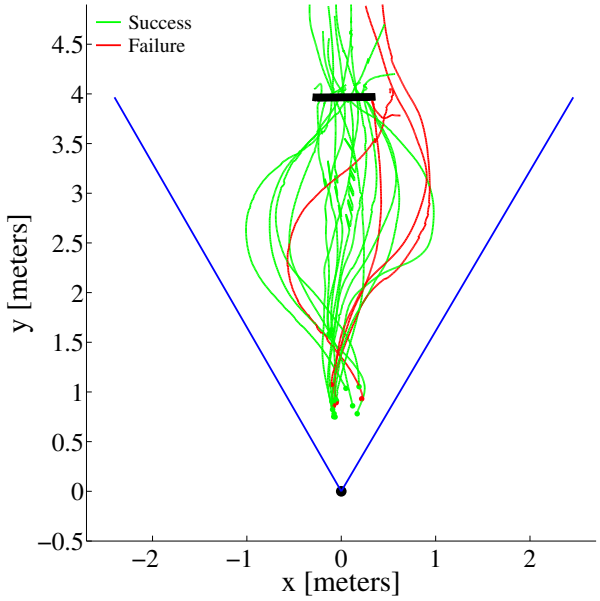

Figure 10: Plot of camera field of view and experimental trials within the feasible region.

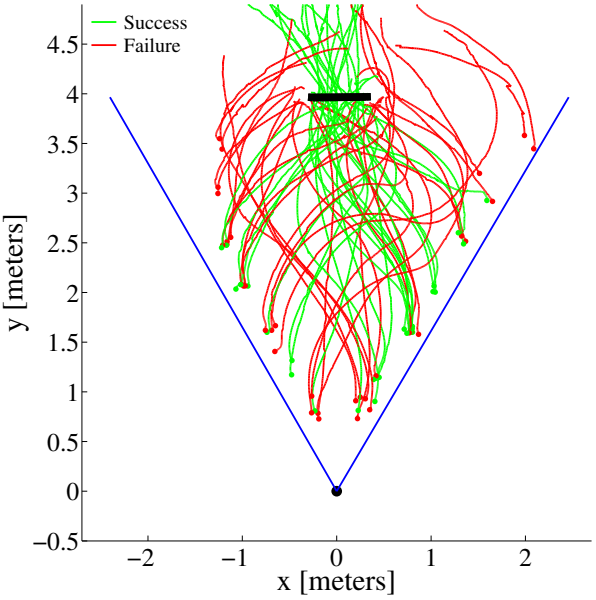

Figure 11: Plot of camera field-of-view and perimeter experimental trials overlayed.

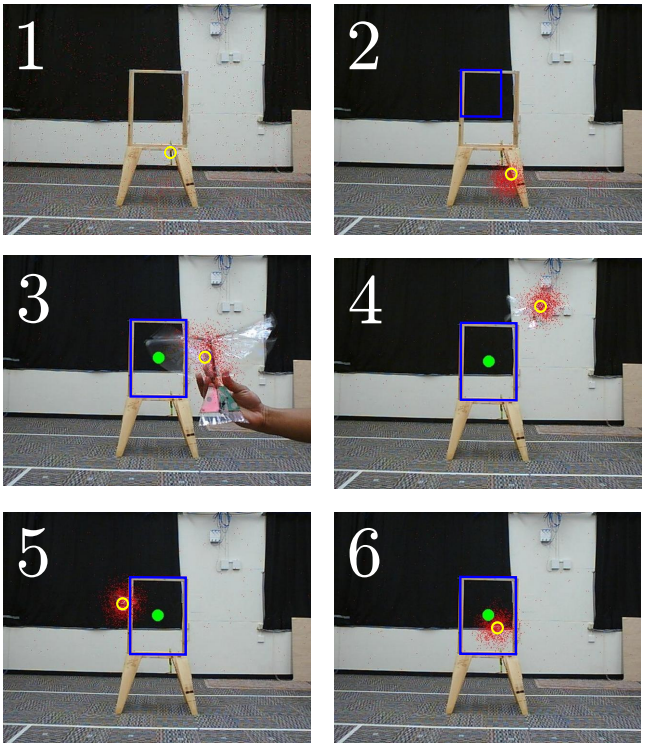

Figure 9: Frame sequence from video feed used for tracking. Particles are shown in red, the current state estimate as a yellow circle, the window bounding box in blue, and the target location as a green circle.

6. CONCLUSIONS AND FUTURE WORK

We have demonstrated cooperative guidance of a 13 gram flapping-wing MAV with a lightweight base station. In addition, we developed a model of the cooperative system using system identification methods on the individual system capabilities, and composing it with known system interactions. The corroboration between our simulated and experimental results shows that our simplified modeling approach is useful for predicting the performance of cooperative systems at low computational cost. Our model could also be used to determine when to begin control with high probability of success, even though we lack information about the complete pose of the MAV.

We intend to further explore the benefits and limitations of collaboration in unstructured environments by deploying our base station on millirobotic ground vehicles. The effectiveness of our modeling approach, when applied to more complex systems with more interactions and agents is also a future area of interest.

7. ACKNOWLEDGMENTS

The authors thank Fernando Garcia Bermudez for his assistance with the Vicon motion capture system, Andrew Pullin for his help with robot photography, and the members of the Biomimetic Millisystems Laboratory and the EECS community at the University of California, Berkeley for their advice and support. This material is based upon work supported by the U.S. Army Research Laboratory under the Micro Autonomous Systems and Technology Collaborative Technology Alliance, and the National Science Foundation under Grant No. IIS-0931463.

8. REFERENCES

[1] M. A. R. Ahad. *Computer Vision and Action Recognition: A Guide for Image Processing and*

Computer Vision Community for Action Understanding. Atlantis Publishing Corporation, 2011.

[2] S. Baek and R. Fearing. Flight forces and altitude regulation of 12 gram i-bird. In *IEEE RAS and EMBS Int.l Conf. on Biomedical Robotics and Biomechatronics*, pages 454–460, Sept. 2010.

[3] S. Baek, F. Garcia Bermudez, and R. Fearing. Flight control for target seeking by 13 gram ornithopter. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pages 2674–2681, Sept. 2011.

[4] B. Cheng and X. Deng. Translational and rotational damping of flapping flight and its dynamics and stability at hovering. *IEEE Trans. on Robotics*, 27(5):849–864, 2011.

[5] G. de Croon, K. de Clercq, R. Ruijsink, B. Remes, and C. de Wagter. Design, aerodynamics, and vision-based control of the DelFly. In *Int. Journal of Micro Air Vehicles*, volume 1, pages 71–97, June 2009.

[6] G. de Croon, E. de Weerdt, C. De Wagter, B. Remes, and R. Ruijsink. The appearance variation cue for obstacle avoidance. *IEEE Trans. on Robotics*, 28(2):529–534, April 2012.

[7] D. Doman, M. Oppenheimer, and D. Sigthorsson. Dynamics and control of a minimally actuated biomimetic vehicle. Part 1: Aerodynamic model. In *AIAA Guidance, Navigation, and Control*, 2009.

[8] A. Doucet, J. De Freitas, and N. Gordon. *Sequential Monte Carlo Methods in Practice*. New York: Springer, 2001.

[9] D. Floreano, J. Zufferey, M. Srinivasan, and C. Ellington. *Flying Insects and Robots*. Springer, 2009.

[10] W. Hamilton and W. Hamilton. *Elements of quaternions*. Longmans, Green, & co., 1866.

[11] J. Hyams, M. Powell, and R. Murphy. Cooperative navigation of micro-rovers using color segmentation. In *IEEE Int. Symp. on Computational Intelligence in Robotics and Automation*, pages 195–201, 1999.

[12] J. S. H. Imraan Faruque. Dipteran insect flight dynamics. Part 2: Lateral-directional motion about hover. *Journal of Theoretical Biology*, 265(3):306–313, 2010.

[13] D. Jung, J. Heinzmann, and A. Zelinksy. Range and pose estimation for visual servoing of a mobile robot. In *IEEE Int. Conf. on Robotics and Automation*, volume 2, pages 1226–1231 vol.2, May 1998.

[14] T. Kanade, O. Amidi, and Q. Ke. Real-time and 3d vision for autonomous small and micro air vehicles. In *IEEE Conf. on Decision and Control*, December 2004.

[15] Z. Khan and S. Agrawal. Control of longitudinal flight dynamics of a flapping-wing micro air vehicle using time-averaged model and differential flatness based controller. In *American Control Conf.*, pages 5284 –5289, July 2007.

[16] D. B. Kingston and A. W. Beard. Real-time attitude and position estimation for small UAVs using low-cost sensors. In *AIAA Unmanned Unlimited Technical Conf., Workshop and Exhibit*, September 2004.

[17] N. Knoebe and T. McLain. Adaptive quaternion control of a miniature tailsitter UAV. In *American Control Conference*, pages 2340–2345, June 2008.

[18] S. M. LaValle. *Planning Algorithms*. Cambridge University Press, Cambridge, U.K., 2006. Available at http://planning.cs.uiuc.edu/.

[19] N. E. Leonard and P. Krishnaprasad. Averaging for attitude control and motion planning. In *IEEE Conf. on Decision and Control*, pages 3098–3104, 1993.

[20] C. Luo, A. Espinosa, A. De Gloria, and R. Sgherri. Air-ground multi-agent robot team coordination. In *IEEE Int. Conf. on Robotics and Automation*, pages 6588–6591, May 2011.

[21] S. Mehta, G. Hu, N. Gans, and W. Dixon. Adaptive vision-based collaborative tracking control of an UGV via a moving airborne camera: A daisy chaining approach. In *IEEE Conf. on Decision and Control*, pages 3867–3872, Dec. 2006.

[22] P. Nebot, D. Gomez, and E. Cervera. Agents for cooperative heterogeneous mobile robotics: a case study. In *IEEE Int. Conf. on Systems, Man and Cybernetics*, volume 1, pages 557–562 vol.1, Oct. 2003.

[23] A. Pullin, N. Kohut, D. Zarrouk, and R. Fearing. Dynamic turning of 13 cm robot comparing tail and differential drive. In *IEEE Int. Conf. on Robotics and Automation*, pages 5086 –5093, May 2012.

[24] P. Rudol, M. Wzorek, G. Conte, and P. Doherty. Micro unmanned aerial vehicle visual servoing for cooperative indoor exploration. In *IEEE Aerospace Conf.*, pages 1–10, March 2008.

[25] L. Schenato, X. Deng, and S. Sastry. Flight control system for a micromechanical flying insect: Architecture and implementation. In *IEEE Int. Conf. on Robotics and Automation*, pages 1641–1646, 2001.

[26] K. S. Shigeoka. Velocity and altitude control of an ornithopter micro aerial vehicle. Master's thesis, University of Utah, 2007.

[27] T. Stirling, J. Roberts, J. Zufferey, and D. Floreano. Indoor navigation with a swarm of flying robots. In *IEEE Int. Conf. on Robotics and Automation*, pages 4641 –4647, may 2012.

[28] S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics*. MIT Press, Cambridge, Mass, 2005.