# Flying Between Obstacles with an Autonomous Knife-Edge Maneuver

Andrew J. Barry, Tim Jenks, Anirudha Majumdar, and Russ Tedrake

***Abstract*— Motivated by the extraordinary control capabilities of birds flying through clutter, we seek to address the open question of whether autonomous fixed-wing aircraft can exhibit similar performance in obstacle-rich environments. We address this question by developing a small autonomous aircraft that is capable of a high speed (10 body-lengths per second) "knife-edge" maneuver through a gap that is smaller than its wingspan. The maneuver consists of flying towards a gap between two obstacles, rolling to a significant angle, accurately navigating between the obstacles, and rolling back to horizontal. We address the necessary hardware, estimation, planning and feedback control for this challenging maneuver. Results from hardware experiments validate the reliability and repeatability of the control and flight systems.**

## I. Introduction

Avian flight far exceeds our best aircraft control systems. Common birds routinely execute maneuvers well outside the bounds of our flight controllers, such as rapidly navigating through a forest, darting through extremely tight spaces, and recovering from large disturbances (Figure 1). Our goal is to understand how to make small fixed-wing aircraft achieve similar feats in equally challenging environments.

In this work, we focus on the hardware, planning, and feedback control problem. We assume that our system is given full sensing information about its location and the environment. The specific task we execute is a "knife-edge" maneuver, in which a 28-inch wingspan aircraft is launched at 7 meters per second (16 MPH) and must execute a dramatic roll to navigate through a gap that is smaller than its wingspan (Figure 2). This task forces our system to roll 70 degrees in under two body-lengths while maintaining precise tracking following a 9-G launch that accelerates the aircraft to 10 body-lengths per second.

## II. Related Work

There has been substantial progress in control for autonomous flying robots in recent years. Mellinger and Kumar developed an impressive array of maneuvers for quadrotor vehicles, from flying through small gaps [13] to cooperative grasping, transport, and formation flight [12]. Muller, Lupashin, and Andrea presented a system for quadrotor vehicles to juggle between themselves in flight [14]. Tomlin's group has demonstrated successful quadrotor blackflips without the use of external motion capture as well as systems to avoid collision between multiple quadrotors [5].

The authors are with the Computer Science and Artificial Intelligence Laboratory (CSAIL) at the Massachusetts Institute of Technology, Cambridge, MA, USA {abarry,tjenks,anirudha,russt}@csail.mit.edu

Fig. 1: Pigeon flying through an obstacle course. Image courtesy of Andrew Biewener, Concord Field Station / Harvard University [8].

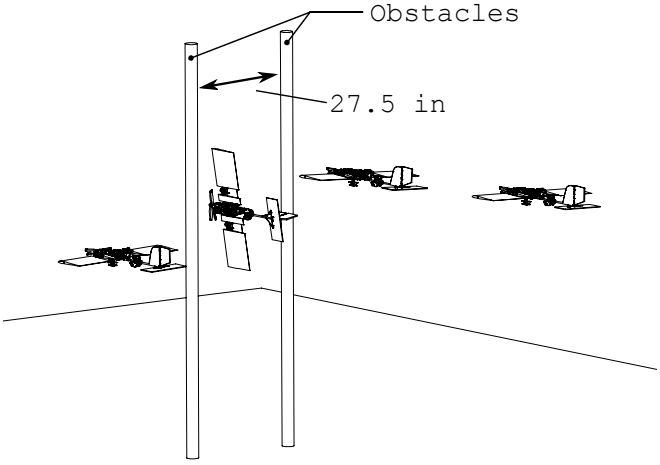



Fig. 2: Sketch of the knife-edge task. The aircraft and obstacles' position and orientation are taken from flight data.

Aerobatic maneuvers on fixed-wing aircraft are generally considered more difficult than on rotorcraft. The quadrotor platforms that have become common in robotics, for example, are able to almost instantaneously apply force along more directions than a traditional fixed-wing, with their ability to hover, turn in place, and accelerate along an almost arbitrary path in three dimensions. In contrast, fixed-wing aircraft must coordinate their control actions, often by changing their orientation to generate lift and/or thrust along the desired vector.

One proposed approach to agile flight control design is via imitation learning. Abbeel used this to great effect to reproduce an acrobatic airshow on his autonomous helicopter [1]. In this work, we are interested in maneuvers with tight

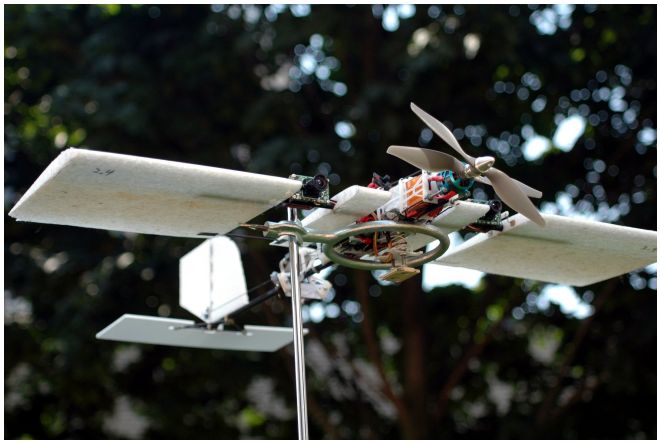


Fig. 3: Experimental platform.

kinematic constraints: our aircraft must perform a perform a 70-degree roll in 0.22 seconds immediately followed by a 60 degree roll in the reverse direction in 0.16 seconds. Furthermore, we are interested in algorithms that will allow planning through previously unknown obstacle fields. Although one cannot be sure, we felt that this class of maneuvers would be difficult for even an extremely skilled pilot.

The work we present here builds more closely on the previous work in fixed wing acrobatics. Cory used a similar approach on a fixed-wing glider to perch on a wire [2]. Sobolic's system uses a similar aircraft model to transition from the hover regime to forward flight [16].

One other key difference in the work we present here is the use of a "wingeron" design that has not previously been explored on agile autonomous aircraft. The design calls for the use of entire wings as control surfaces and offers extremely high roll-rates at the cost of slightly more complex dynamics.

## III. Aircraft Hardware

We have built an unmanned aerial vehicle (UAV) research platform to serve as the testbed for algorithms and as the final metric of performance (Figure 3). We use a "wingeron" design that does away with the traditional wing and aileron seen on most aircraft in favor of a wing that is completely actuated; in other words, the entire wing rotates to act as a control surface. This greatly expands the aircraft's roll ability, only saturating the roll rate when the wingeron approaches $90^o$ to the oncoming flow. The wingerons, however, complicate the flight dynamics due to the additional possibility of a single wing stall during a turn. For example, if the aircraft is executing a right roll, the left wingeron will be deflected up to generate additional lift. Should that wingeron deflect too much, it will stall, causing a loss of lift and the possibility of the aircraft rolling in the opposite direction.

We use an asymmetric wing optimized using XFOIL [4] for flight at 5-15 meters per second (Figure 4). We cut the wing out of expanded polypropylene (EPP) foam, chosen for its ability to absorb kinetic impacts without

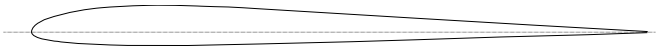


Fig. 4: Airfoil profile, designed to optimize lift at speeds of 5-15 m/s.

permanent deformation. The wings are cut with a computer-controlled hot wire to produce the appropriate airfoil and then strengthened with light-weave fiberglass on the trailing three quarters to ensure that the wing does not significantly deform when subjected to torsional stress.

For propulsion, we use small outrunner brushless DC motors and APC propellers commonly found on aircraft of this scale. We use a two-motor, counter-rotating propeller design that reduces the torque on the airframe produced by changing the throttle. For only a minor efficiency cost, we are able to effectively eliminate this effect, rendering the roll dynamics invariant to the time derivative of the throttle.

Our on-board electronics package is based around a ARM Cortex-A8 running Linux connected to an Atmel microcontroller that is in turn connected to the motor speed controllers, actuator servo motors, 3 axis gyroscope, 3-axis accelerometer, and 3-axis magnetometer.

Due to the limited space in our motion-capture environment, we require a system to accelerate the plane to full speed very rapidly. We use stretched elastic to accelerate a carriage down an approximately meter-long (3.2 ft) rail, with a 9-G peak force. A bar in front of the aircraft holds the plane in place, even at full throttle, allowing us to release the aircraft while the motors are running. The carriage is tall enough to give clearance for the propellers.

## IV. Control

We denote the aircraft's body-centric position in 3-dimensional space and characterize the aircraft dynamics, $\dot{x} = f(x, u)$, using a 12-dimensional state as follows:

$$x = \begin{bmatrix} x_p & y_p & z_p & \psi & \theta & \phi & U & V & W & R & Q & P \end{bmatrix}^T$$

$x_p$, $y_p$, and $z_p$ denote the aircraft position in the world coordinate frame. $\psi$, $\theta$, and $\phi$ denote yaw, pitch, and roll respectively. $U$, $V$, and $W$ are the derivatives of the positions expressed in body-frame coordinates and $R$, $Q$, and $P$ are the aircraft angle derivatives expressed in body-frame coordinates.

We control five inputs on the aircraft: throttle and the deflections of the elevator, rudder, left wingeron, and right wingeron. We approximate these inputs as instantaneous to avoid adding additional states to our system.

$$u = \begin{bmatrix} \text{throttle} \\ \text{elevator} \\ \text{rudder} \\ \text{wingeron\_left} \\ \text{wingeron\_right} \end{bmatrix}$$

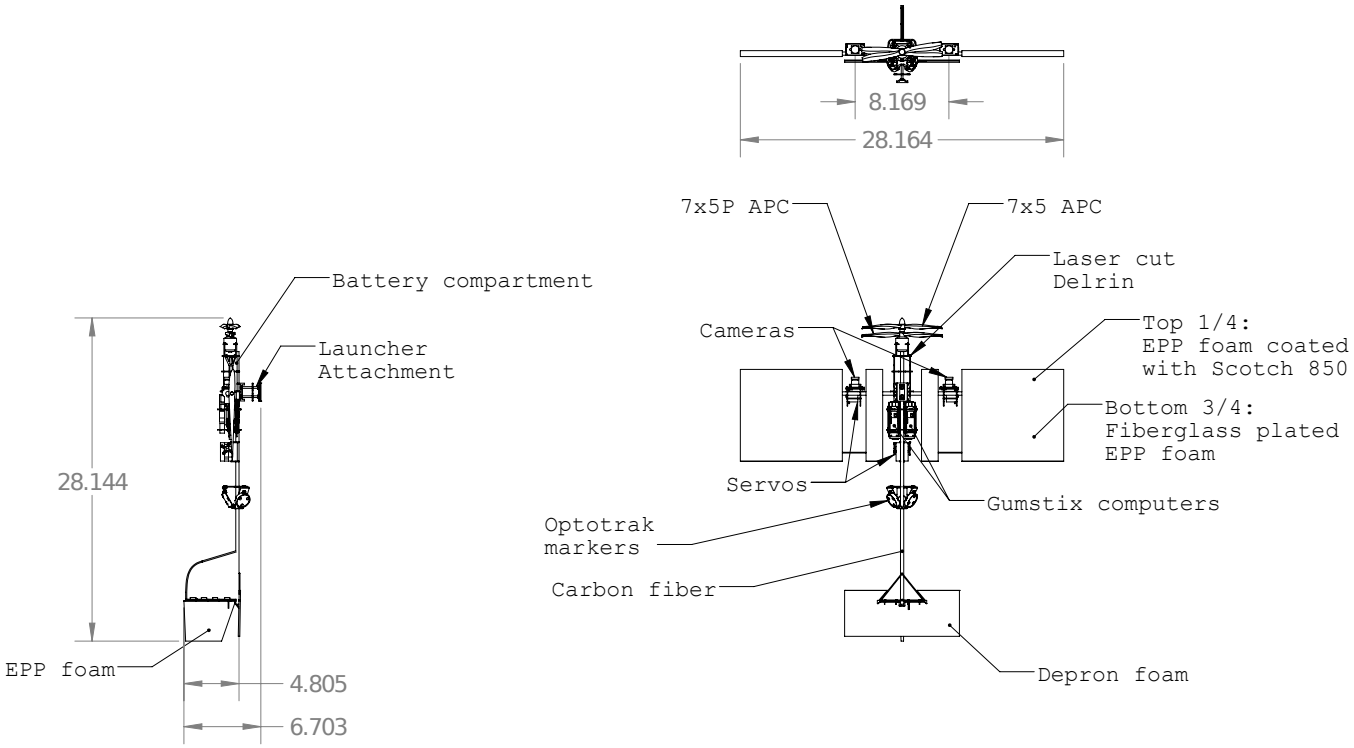



Fig. 5: Scaled and annotated drawing of our research platform. All dimensions are in inches.

### A. Aircraft Model

We use a model-based formulation for both planning and control, so we require a high-fidelity model of our aircraft. Stevens and Lewis [17] and Sobolic [16] provide proven aircraft models that we use as a basis. To build our model, we modified the standard aircraft model in [17] to account for our wingeron design and split wings. Since the wings are independently actuated, we compute their angles of attack separately. Each wing's deflection is added to the body's angle of attack and then a piecewise polynomial function maps that angle to the appropriate lift and drag coefficients. We compute these coefficients using an XFOIL analysis of the wing inside the laminar flow regime and flat plate theory outside that realm.

Ideally, one could fit the parameters of the aerodynamic model using only the airfoil shape, the CAD model, and measurements on the airframe, but it is difficult to obtain a good estimate of the drag forces on the body. To identify these parameters, we fly the aircraft repeatedly over a range of conditions and fit the following: prop-wash velocity, thrust, body component of drag, and body component of drag in the vertical axis (Z). We use the prediction error minimization method implemented in the MATLAB System Identification Toolbox [9] for these estimates.

### B. Open-Loop Planning

To create a feasible plan for an aircraft, we must ensure that the desired trajectory generates enough lift, does not saturate the control surfaces, and is aware of the changing system models during stall. The state space of a 3D aircraft is large (at least 12-dimensional), so exhaustive search or even dynamic programming methods are prohibitively expensive [3]. To mitigate this issue, we resort to locally optimal methods that can optimize a given trajectory to satisfy dynamical constraints and minimize cost but cannot provide guarantees of global optimality. We use a direct collocation method that optimizes trajectories by solving a nonlinear programming problem with the system's dynamics added as constraints to the optimization. This method requires the optimizer to minimize both the cost function and to find a trajectory that satisfies the dynamics [18].

We utilize a standard direct collocation method as given by Hargraves and Paris [6] for trajectory generation and optimization. The method uses cubic polynomials to represent the system's trajectory and collocation to satisfy the dynamics.

To generate trajectories, we begin with a tape that starts at $x_0$ and flies straight to the goal, $x_f$. We then minimize the cost over control actions while ensuring that the dynamics are satisfied and that we do not impact an obstacle:

$$\begin{aligned} \min_{u,x} \quad & h(t_f) + \int_0^{t_f} g(x(t), u(t))dt, \\ s.t. \quad & x(0) = x_0, \\ & x(t_f) = x_f, \\ & \dot{x} = f\left(x(t), u(t)\right), \\ & \phi_i(x(t)) < 0, \forall i \end{aligned}$$

where $f$ is the dynamics and $\phi_i$ is the constraint on obstacles

as defined below.

Our cost function focuses on the control actions of the aircraft. The one-step cost is $g = u^T Ru$, where $u$ is the control vector introduced earlier. We let the system's final cost simply equal to the final time, promoting solutions requiring less time in the air: $h = t_f$.

To ensure that our trajectory does not collide with obstacles, we add constraints based on the distance to each obstacle. With the cost function and constraints, we simultaneously balance avoiding obstacles and limiting control actions. For the purposes of this trajectory, we use cylindrical obstacles that are infinitely tall and defined to be along the Z-axis.

We compute the distance from the vehicle to the obstacle by projecting both the aircraft and obstacle's image onto the XY-plane. In this projection, we approximate the aircraft's shape as a rectangle, taking the yaw, pitch, and roll angles into account. This projection results in our aircraft becoming "thinner" on the XY-plane as it rolls, promoting a high roll angle when maneuvering through the obstacles.

If we let $d$ be the minimum distance from the projected rectangle to the center of the obstacle's projected circle, we can use the following function for the constraint:

$$\phi_i = \tanh(r - d)$$

where $r$ is the radius of the obstacle. Figure 6 visualizes the constraint function at different roll angles.

### C. Feedback Control with a Time-Varying Linear Quadratic Regulator

Inspired by the recent success with time-varying linear quadratic regulators (TVLQR) such as [1] and [2], we use this method to perform feedback around our desired trajectory. In this formulation, we use a standard linear quadratic regulator (LQR) controller, but move the goal point at each instant in time, linearizing around the reference trajectory [15]. The algorithm is closely related to differential dynamic programming [11] with similar formulations for costs and results for linear systems.

Given a nominal trajectory, $x_0(t)$, and a nominal control input, $u_0(t)$, we can define new coordinates centered around $x_0(t)$ as follows:

$$\bar{x}(t) = x(t) - x_0(t), \quad \bar{u}(t) = u(t) - u_0(t)$$

Linearizing the time-varying dynamics of $\bar{x}(t)$, we obtain:

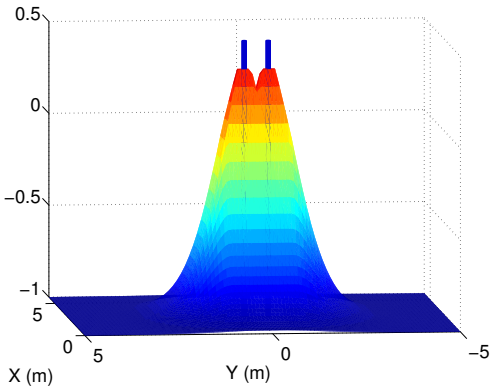
$$\dot{\bar{x}}(t) \approx A(t)\bar{x}(t) + B(t)\bar{u}(t).$$

The control law is obtained by solving a Riccati differential equation:

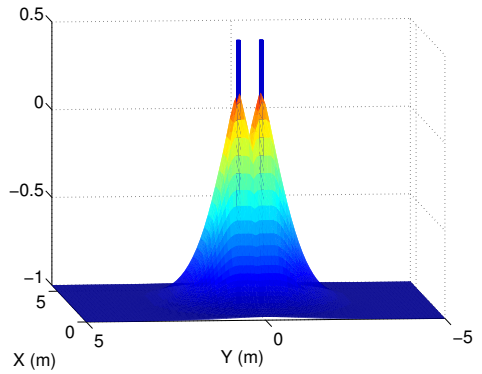
$$-\dot{S}(t) = Q - S(t)B(t)R^{-1}B^T S(t) + S(t)A(t) + A(t)^T S(t)$$

with final value conditions $S(t) = S_f$. Here, $Q$ and $R$ are positive-definite cost-matrices. The feedback control law is then given by:

$$\bar{u}(x,t) = -R^{-1}B^T(t)S(t)\bar{x}(t) = -K(t)\bar{x}(t).$$




(a) Roll set to $0^o$.




(b) Roll set to $70^o$.

Fig. 6: Constraint functions for two obstacles and different roll configurations as a 2D slice of space (all states other than $x$, $y$, and $\phi$ are set to zero). Obstacles are shown as blue cylinders. The optimization is constrained to allow the aircraft only to pass through locations where the values are less than zero. A $\tanh()$ function is used to give the optimizer a gradient that is easy to follow.

The TVLQR controller requires some hand-tuning of the cost matrix before it will track the open-loop trajectory adequately. We performed this hand-tuning based on repeated flights and comparisons between the system's actual and desired paths. The costs used in the knife-edge task are as follows: $Q = diag([100, 100, 100, 1000, 100, 100, 10, 10, 10, 42, 10, 10])$, $R = diag([0.2, 2, 2, 0.6, 0.6])$.

### D. State Estimator

Our motion capture system reports the pose of the plane at 70 $Hz$. We use finite differences combined with a Luenberger observer [10] to obtain estimates of the derivatives of the positions and angles expressed in the body frame. Denoting our dynamics model by $\dot{x} = f(x,u)$, we can write the dynamics of the estimated state, $\hat{x}$, as:

$$\dot{\hat{x}} = f(\hat{x}, u) + L(y - \hat{x})$$

where, $y$ denotes the observation and $L$ is the hand-tuned Luenberger gain. $y$ consists of pose estimates reported by the

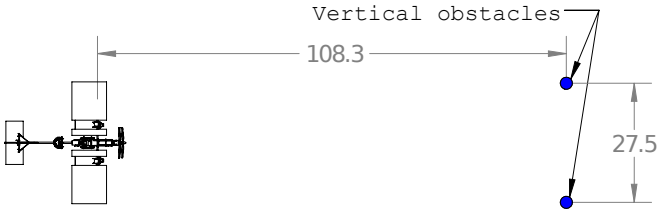



Fig. 7: Diagram of the experimental setup (to scale). All dimensions are in inches.

motion capture system and derivatives estimated from finite differencing. At each time-step, an Euler update is used to obtain the current estimated state:

$$\hat{x}[t] = \hat{x}[t - dt] + \dot{\hat{x}}[t]dt.$$

Since the pose estimates reported by the motion capture system are reliable, we ignore the pose variables in the estimated state and keep only the derivative variables. During the short periods of time when the motion capture markers are occluded by obstacles, we estimate the state by forward simulating the aircraft model.

## V. Results

Our experiment consists of two poles 0.7 meters (27.5 inches) apart and 2.75 meters (9 feet) in front of our launching system. Figure 7 gives an scaled drawing of the setup.

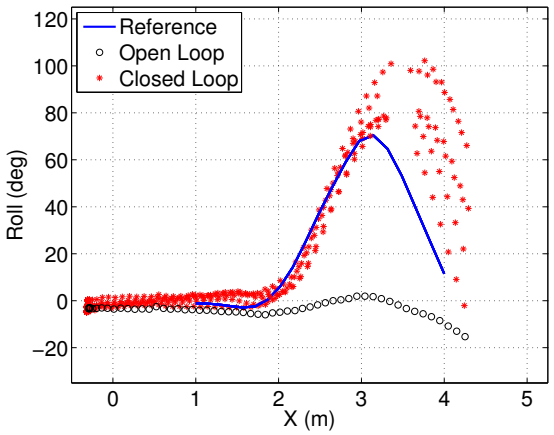
The hardware system has proven to have impressive performance and to be robust to multiple failures. We have flown hundreds of test flights, each ending with an impact into a net, with only minor, repairable hardware failures. The airframe is robust to conditions created from sensing, control, and computational failure, as well as loss of power and unintended impact with the launching system. None of these minor repairs required changing model parameters.

We note that the launching mechanism produced consistent initial conditions, simplifying the modeling and tuning problems. This allowed us to rely on the launcher to place the aircraft in flight in states near where the open-loop trajectory started.

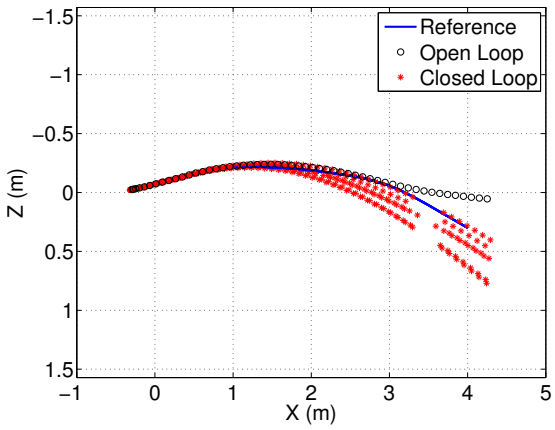
We find that the system's tracking is sufficient for repeated flights through the obstacle field without collision. We have flown 9 flights through the vertical obstacles using two different locally optimal trajectories. On every flight the aircraft successfully navigated the gap without collision.

Figure 8 shows the results of six repeated flights with this system, the desired trajectory, and the result of running the system without feedback. Figure 10 gives snapshots of the aircraft performing this trajectory in our test environment.

It is interesting to note that the feedback system primarily corrects for errors in the models. Figure 9 shows action-tapes for six closed-loop executions and the open-loop reference for the control surfaces. The closed-loop tapes are consistent, but they differ from the open-loop tape, indicating that the feedback system is working to correct errors in modeling.




(a) Aircraft flight data for roll vs. forward flight. Red shows flights with the closed-loop controller running and black shows flights running only the open-loop tape.



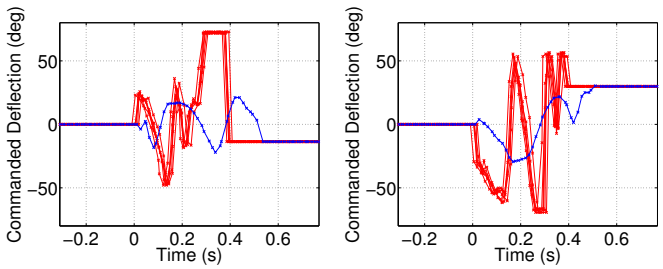

(b) Comparison of forward flight (X-axis) vs. height (Z-axis). Note that the Z axis is plotted in reverse to give a more intuitive view of the aircraft's height (in our coordinate system increasing Z moves towards the ground).

Fig. 8: Motion capture data from the aircraft in flight compared with the planned knife-edge maneuver. We show both open-loop (black) and closed-loop (red) flights. The gaps in the data occur when the plane's markers pass by the obstacles, causing the motion capture system to lose sight of the plane for a small period of time.

### A. Extension to Horizontal Obstacles

The knife-edge maneuver demonstrates good tracking and system performance but does not require the system to perform more than one maneuver during a flight. The same system should be able to perform multiple maneuvers. To test this, we added two horizontal obstacles after the two vertical obstacles. This configuration requires the aircraft to quickly roll back to level flight and maintain altitude tracking to succeed in avoiding both obstacles.

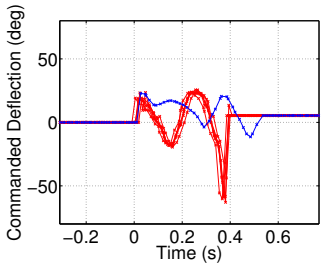
We used our existing model to plan a new trajectory with horizontal obstacles, adding constraints similar to those used for the vertical barriers. We then tuned the LQR costs for that trajectory to improve tracking. Figure 12 shows snapshots of the flight and Figure 13 show images from the on-board



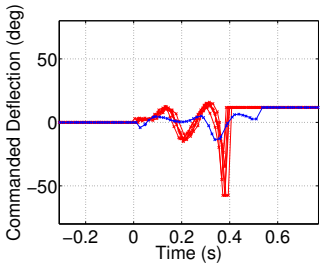

(a) Left wingeron




(b) Right wingeron




(c) Elevator




(d) Rudder

Fig. 9: Comparison between open-loop control tapes (blue) and closed-loop tapes (red) for the knife-edge trajectory. Note the similarity between six successful runs of the knife-edge trajectory and the dissimilarity between those and the open-loop tape. This indicates that our feedback system is primarily correcting for model inaccuracies.

camera during flight.

## VI. Future Work and Conclusion

### A. FPGA Vision and Outdoor Navigation

Recent work on FPGA vision systems has provided encouraging results that our platform may be able to collect and process information about its surroundings at 120 frames per second [7]. We are considering moving our platform out of a motion capture environment, adding an FPGA processor, and determining where obstacles are in real time.

Moving out of a motion capture environment will require a substantial sensing and estimation effort, which will extend our sensors to airspeed, barometric altitude, GPS, and others. Our aircraft is capable of carrying these sensors and we look forward to UAVs that can fly at high speed, identifying and avoiding obstacles faster than any human pilot.

### B. Conclusion

We have demonstrated a system that is capable of performing a knife-edge maneuver, rolling 70 degrees in under two body-lengths, while moving at over 10 body-lengths per second. Our system, using an aerodynamic model, direct collocation based trajectory optimization, and TVLQR is capable of performing the maneuver robustly. Finally, we extend the work to a more challenging obstacle field and show that the techniques are sufficient for that task as well.

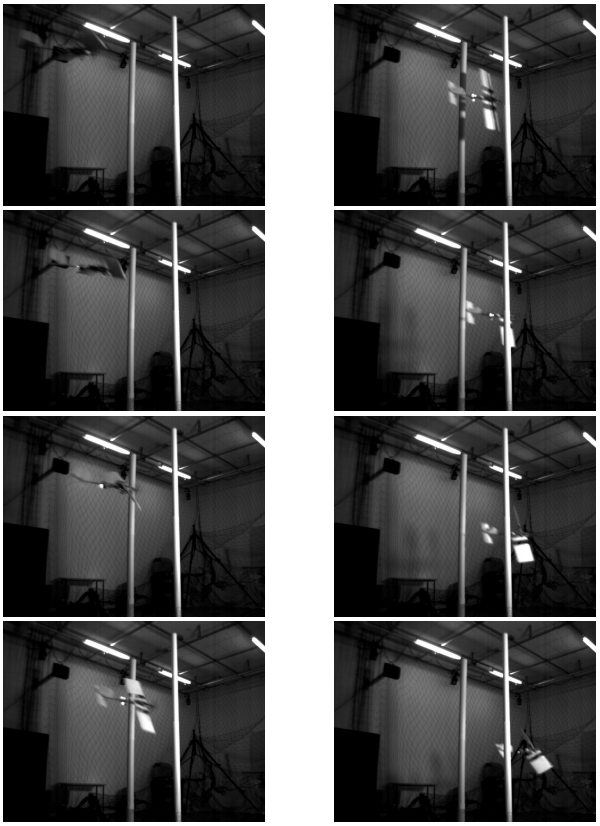


Fig. 10: Sequence of stills from a knife-edge maneuver. Each image is 0.0417 seconds (41.7ms) after the previous. The entire set captures a total of 0.292 seconds. The full flight, including onboard video, is shown in the video attachment.

Fig. 11: Sequence of stills from an onboard camera during the knife-edge maneuver. Each image is 0.083 seconds (83ms) after the previous. The entire set captures a total of 0.583 seconds.

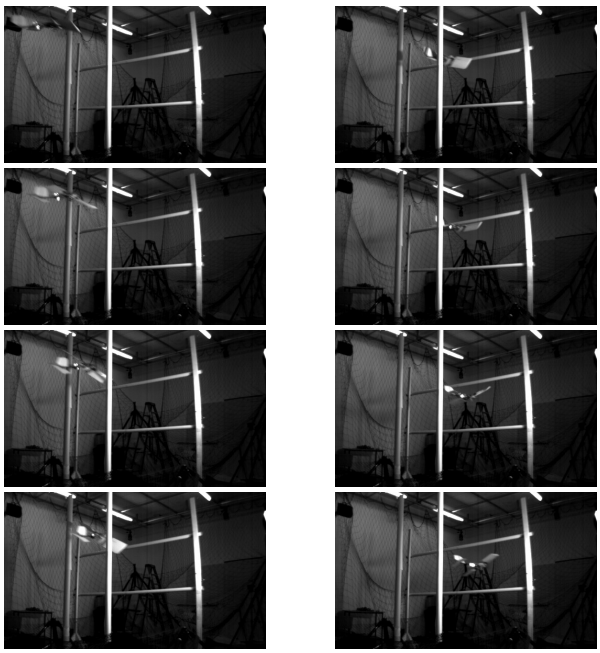


Fig. 12: Sequence of stills from a four-obstacle maneuver. As before, each image is 0.0417 seconds (41.7ms) after the previous. The entire set captures a total of 0.292 seconds.

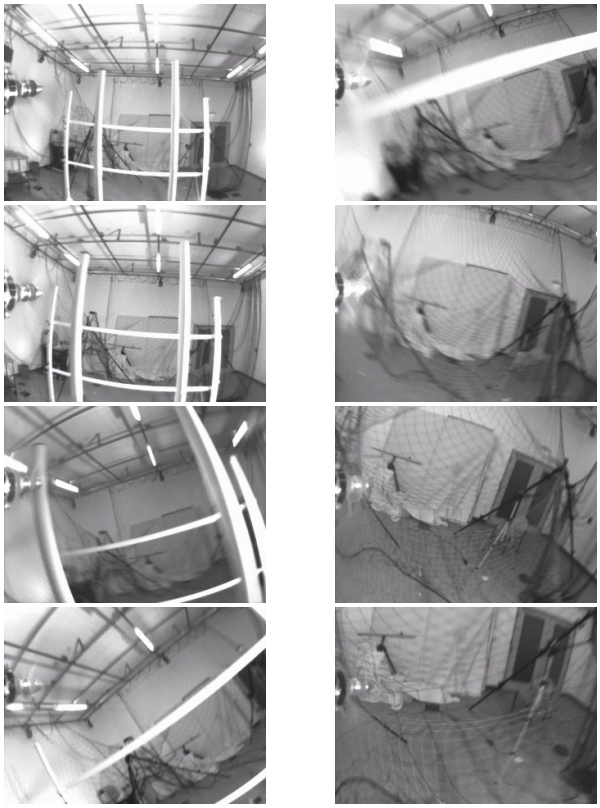


Fig. 13: Sequence of stills from an onboard camera during the four-obstacle knife-edge maneuver. Each image is 0.083 seconds (83ms) after the previous. The entire set captures a total of 0.583 seconds.

## VII. Acknowledgements

This work was supported by ONR MURI grant N00014-09-1-1051. Andrew Barry is partially supported by a National Science Foundation Graduate Research Fellowship. Anirudha Majumdar is partially supported by the Siebel Scholars Foundation.

## References

[1] P. Abbeel, A. Coates, M. Quigley, and A. Y. Ng. An application of reinforcement learning to aerobatic helicopter flight. In *Proceedings of the Neural Information Processing Systems (NIPS '07)*, volume 19, December 2006.

[2] R. Cory. *Supermaneuverable Perching*. PhD thesis, Massachusetts Institute of Technology, June 2010.

[3] M. Diehl, H. Bock, H. Diedam, and P. Wieber. Fast direct multiple shooting algorithms for optimal robot control. *Fast Motions in Biomechanics and Robotics*, pages 65–93, 2006.

[4] M. Drela and H. Youngren. Xfoil 6.94 user guide. 2001.

[5] J. H. Gillula, G. M. Hoffmann, H. Huang, M. P. Vitus, and C. Tomlin. Applications of hybrid reachability analysis to robotic aerial vehicles. *The International Journal of Robotics Research*, 2011.

[6] C. R. Hargraves and S. W. Paris. Direct trajectory optimization using nonlinear programming and collocation. *J Guidance*, 10(4):338–342, July-August 1987.

[7] D. Honegger, P. Greisen, L. Meier, P. Tanskanen, and M. Pollefeys. Real-time velocity estimation based on optical flow and disparity matching. *To appear, Intelligent Robots and Systems (IROS) 2012*, Oct. 2012.

[8] H. Lin, I. Ros, and A. Biewener. Through the eyes of a bird: A vision-based model for pigeon obstacle avoidance flights. In preparation.

[9] L. Ljung. System identification toolbox for use with matlab. 2007.

[10] D. Luenberger. An introduction to observers. *Automatic Control, IEEE Transactions on*, 16(6):596–602, 1971.

[11] B. Mayne. A second-order gradient method for determining optimal trajectories of non-linear discrete-time systems. *International Journal of Control*, 3(1):85–95, 1966.

[12] Mellinger, D., Shomin, M., Michael, N., Kumar, and V. Cooperative grasping and transport using multiple quadrotors. In *Proceedings of the international symposium on distributed autonomous robotic systems*, 2010.

[13] D. Mellinger, N. Michael, and V. Kumar. Trajectory generation and control for precise aggressive maneuvers with quadrotors. In *Proceedings of the 12th International Symposium on Experimental Robotics (ISER 2010)*, 2010.

[14] M. Muller, S. Lupashin, and R. D'Andrea. Quadrocopter ball juggling. In *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, pages 5113–5120. IEEE, 2011.

[15] P. Reist and R. Tedrake. Simulation-based LQR-trees with input and state constraints. In *Proceedings of the International Conference on Robotics and Automation (ICRA)*, 2010.

[16] F. M. Sobolic. Agile flight control techniques for a fixed-wing aircraft. Master's thesis, Massachusetts Institute of Technology, June 2009.

[17] B. Stevens and F. Lewis. *Aircraft Control and Simulation*. John Wiley & Sons, Inc., 1992.

[18] O. von Stryk. Numerical solution of optimal control problems by direct collocation. In *Optimal Control, (International Series in Numerical Mathematics 111)*, pages 129–143, 1993.