Controller design for a wall landing airplane

Alexis Lussier Desbiens, Etienne Le Grand
{alexisld,legrand}@stanford.edu

December 11, 2009

***Abstract* – Plane perching on a wire has been demonstrated with success in laboratory conditions by [Cory & Tedrake, 2008]. The present paper focuses on designing a controller for a small wall landing glider. We use a simulator and a reinforcement learning algorithm in a coarsely discretized state-space to design an optimal control for the landing. Analysis of this policy allows us to deduce a simpler closed loop controller that is robust to variations in initial speed and that only relies on pitch and pitch rate measurement. A controller of that class is now being implemented on the real airplane.**

Problem Description

The Biomimetics & Dextrous Manipulation Laboratory (BDML) at Stanford has been working on allowing perching on vertical surfaces for small UAVs. The maneuver usually goes like this: the plane flies toward the wall at cruise speed (10-14 m/s), detects the wall at 6m, pitches up to 90 degrees to rapidly decrease its speed by increase drag and touches down at about 1-3 m/s. A specially designed suspension then absorbs the impact and allows the micro-spines (claws) to engage the wall. When properly executed, this maneuver is quite robust, always maintains some forward speed (and thus controllability) and the suspension provides a large region of velocities and orientations at which touchdown is possible. This envelope of velocities and orientations is defined as:

$$pitch \in [45°, 110°], \dot{x} \in [0,3] m/s, \dot{y} \in [-2,1] m/s$$

Unfortunately, the current maneuver is sensitive to the initial airspeed of the plane. For the same elevator deflection, the faster the plane goes the larger the pitching moment created will be. The previous feedforward controller was only tuned for a specific airspeed and going faster meant doing a looping, while going slower meant not pitching up enough and crashing onto the wall. Although it would be possible to sense the airspeed by adding a sensor, this would add weight to an already heavy system.

To solve that problem, we first developed a controller able to cope with a wide range of initial speeds using reinforcement learning to design an optimal feedback control (full states controller) on a simulation model of our airplane. Second, we used this simulator to evaluate the initial flying conditions at which the RL controller can successfully bring the plane in the allowable touchdown envelope. Third, we observed the optimal trajectories and deduced a simple controller that is robust to variation of initial airspeed and only uses the sensors currently on the airplane.

Simulator

The simulation model of the glider, inspired by work from [Cory & Tedrake, 2008], had already been developed at the BDML for a glider. This model considers the wing and elevator as flat plates and thus, the lift and drag coefficient can be expressed as a function of the angle of attack (α):

$$C_L = \sin(\alpha)\cos(\alpha) \quad C_D = 2\sin^2(\alpha)$$

A simple propeller model has been added to simulate the prop-wash on the control surfaces and the variable thrust with respect to flying speed during normal flight. The thrust of the propeller can be expressed with a simple approximated relationship:

$$T = T_0 + T_1 v_{incoming}$$

Where T0 is the static thrust and T1 is a negative coefficient that reduces the thrust as the plane velocity increases. Given the thrust of the airplane, it is possible to use the propeller disk theory to calculate the prop-wash on the control surfaces:

$$v_{out} = \sqrt{\frac{2T}{\rho A_{disk}} + v_{incoming}^2}$$

Where ρ is the air density and A_{disk} is the propeller disk area. Note that although the thrust was added to the model, the plane was flown as a glider for the work presented in this paper.

RL Algorithm

Like in [Cory & Tedrake, 2008], we formulate our problem as an infinite-horizon optimal feedback control problem. Our system has a 6 dimensional state space $S = \{(x, y, \dot{x}, \dot{y}, \theta, \dot{\theta})\}$ and a space of actions $A = \{(q_e, T)\}$ with q_e being the elevator command and T the throttle command. For each state s and each command a, the simulator gives a transition function $z(s, a, \delta t) = s'$. Our controller has to output $a^*(s)$ such that if $a(t) = a^*(s(t))$ during the maneuver, the plane will arrive close to the goal $s_g = (x_g, y_g, \dot{x}_g, \dot{y}_g, \theta_g, \dot{\theta}_g)$. In our case, we define s_g as (10m, 0m, 1m/s, -0.5m/s, 90deg, 0deg/s). In order to achieve this control, we define recursively a cost function J with

$$J_0(s) = \min((s - s_g)^T Q (s - s_g), J_{\max})$$

$$J_{i+1}(s) = \min_{a \in A}(J_i(s), J(z(s,a))))$$

Where

$$Q = diag(Q_x, Q_y, Q_{\dot{x}}, Q_{\dot{y}}, Q_\theta, Q_{\dot{\theta}})$$

characterizes the cost of being off the goal in each direction. Note that we make sure that J_0 grows rapidly in the x direction so that the airplane reaches the other optimal touchdown conditions as it touches the wall.

By discretizing the state space, we use a value iteration algorithm to make J converge to J^*. In order to be computationally tractable, we only use 10 bins in each dimension. The convergence is accelerated by the use of an upper bound J_0 and a wise choice of the scanning direction. Instead of discretizing the time, we define a transition $z(s, a)$ as the trajectory of the plane between the state s and the moment when the plane crosses one hyperplane of the discretized space, as shown on the figure 1 in a 2 dimensional case. Also, we use the fact that the transitions don't depend on the coordinates x and y when the discretization step is constant on those axis. Hence, only 10^5 different simulations are actually computed to find the whole set $z(s, a)$ in the case of a glider.

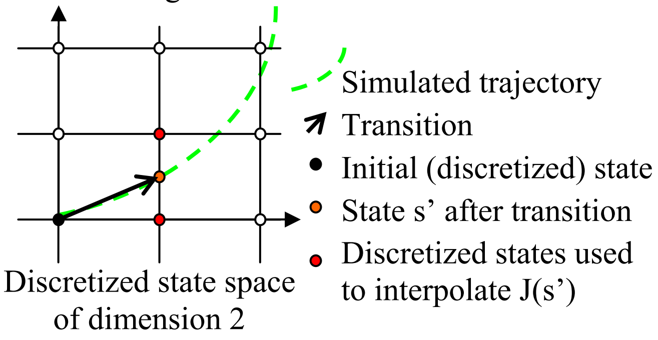
Simulated trajectory
Transition
Initial (discretized) state
State s' after transition
Discretized states used to interpolate J(s')
Discretized state space of dimension 2

Fig 1. Transition bounds, 2 dimensional case

The optimal control is obtained by

$$a^*(s) = \arg\min_{a \in A}(J^*(z(s,a)))$$

Because of the coarse discretization, we cannot be sure that this is a near-optimal control. In order to validate it, we feed it back into the simulator along with several different initial states.

Thanks to an optimized algorithm, we were able to run the value iteration algorithm on the 10^6 states in about 30 seconds. Several discretizations (up to 10^7 states) and cost function initializations have been implemented, all leading to slightly different trajectories. The results presented below come from the standard discretization of 10 bins in each direction with a cost function that was judged suitable for the needs of landing on a wall.

RL Optimal Trajectories

Figure 2 illustrates the results from using the RL controller in our simulator for initial velocities between 10 (red) and 14 m/s (dark blue) on a glider. On this figure, we can see that at the lower initial velocities the plane pitches up at a slower rate. It also doesn't pitch up all the way to 90

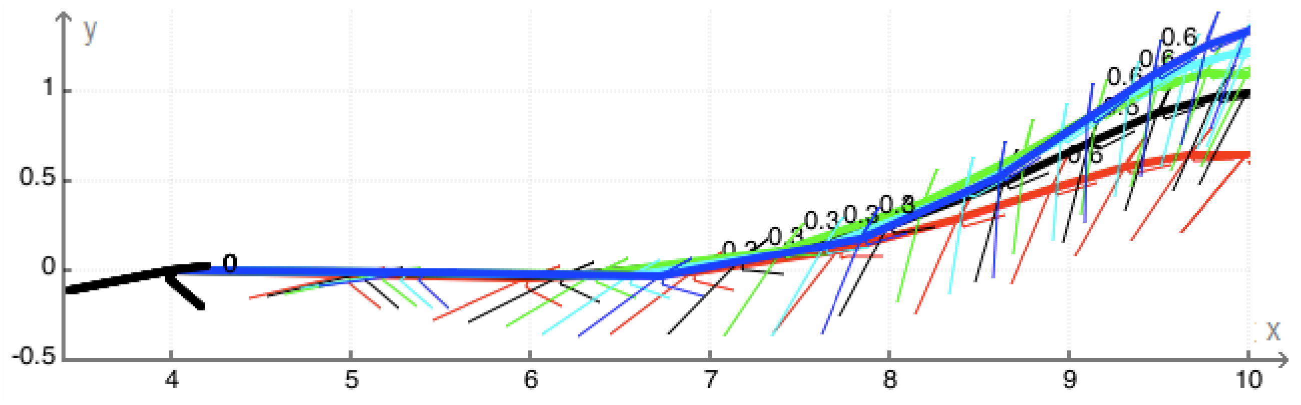

Fig 2. - Trajectories obtained with the RL control policy for different initial speeds

degrees in order to maintain some lift and reach the wall, but pitches up enough to allow the suspension to contact the wall in a proper state. At higher initial velocities, the airplane rapidly pitches up to 90 degrees to shed some airspeed before touchdown. For the initial velocities tested, the RL generated trajectories reach the wall with vx ≈ 3 m/s, vy ≈ -0.5 m/s and a pitch between 45 and 60 degrees.

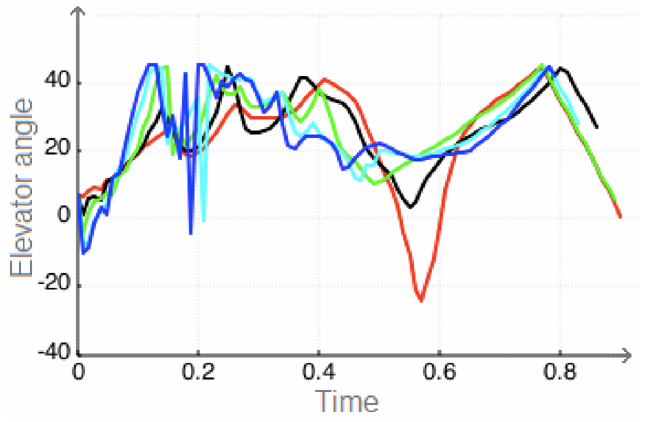

Fig 3. RL generated control

The control input on the elevator for these different trajectories is illustrated on figure 3. We can see that most control inputs initially *kick* the elevator to 45 degrees to initiate the pitch up maneuver, and then maintain the elevator to 25-40 degrees to sustain the rotation before dropping back to -20 to 20 degrees to slow down the rotation of the airplane.

One interesting characteristic of these trajectories, as we can see on figure 4, is that they maintain a constant pitch rate from t=0.1 second up to the point when the desired maximum pitch is reached. Furthermore, although the pitch rate is constant for a specific trajectory, it varies according to the initial speed: when the initial velocity is higher, the glider pitches up faster in order to complete the maneuver in a shorter amount of time.

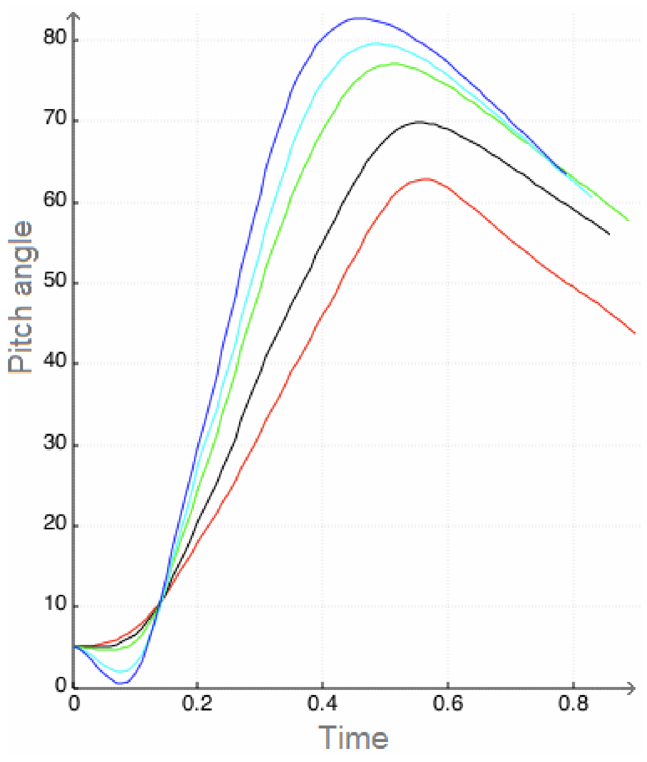

Fig 4. Pitch angle during the landing

Although the RL showed us good trajectories and controller input for a range of initial speeds, this control scheme is difficult to implement on the real airplane for two main reasons: the look-up table for the controller contains 10^6 entries and the airplane doesn't have a full knowledge of its states. Although it might be possible to approximate the controller look-up table by a lower dimension function, it would be hard to add enough sensors without getting excessively heavy. On the current version of the airplane, only attitude and acceleration measurements are readily available. The distance from the wall is briefly known as the sensor has a maximum range of 7m and becomes useless as the plane pitches up to more than 45 degrees (the ultrasound not being reflected anymore).

Simple Controller

To design a simple controller that will only rely on the measurements available on the airplane, it is required to change the dependence of the desired pitch rate value on the initial velocities to something that is related to the initial velocity and that can be measured on the airplane. One way to do this is to initially use a short step function on the elevator and measure the change in angular velocity.

As we said before, the moment created (and thus the angular acceleration) depends on the airspeed over the control surface. As the airplane only has rate gyros, measuring acceleration would require differentiation and we preferred to use the change in angular velocity (Δω) as a measure of the airspeed.

As shown on figure 5, when the optimal pitch trajectories θ(t) are normalized by Δω, the trajectories collapse to a single trajectory with various final pitch exit points related to the measured Δω. Thus, a simple controller can be characterized by:

1. Flying toward the wall until detection (at 6m)
2. Commanding an elevator step output of 45 degrees during about 30 ms and measure the change in angular velocity (Δω).
3. Tracking the θ(t)/Δω optimal trajectory as found through RL (and fine tuned on the real system) with a PD (Proportional-Derivative) controller.
4. Stop the tracking and maintain a small angular velocity once the desired pitch angle is reached (according to the Δω measurement).

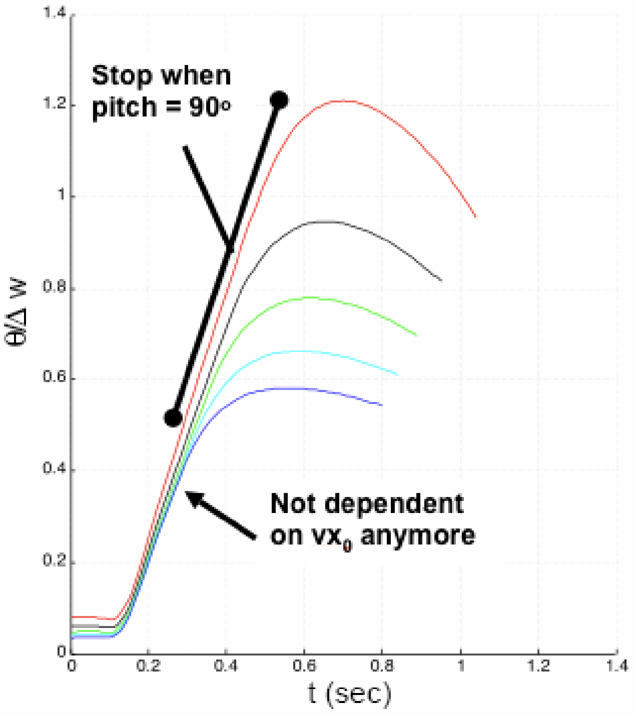

Fig 5. Normalized pitch angles

The result of this algorithm can be seen on figure 6 which shows that a simple PD controller can easily maintain the pitch rate constant during the maneuver and bring the airplane to the desired pitch angle. The overshoot observed represents a tradeoff with a fast rise time and doesn't critically affect the performances.

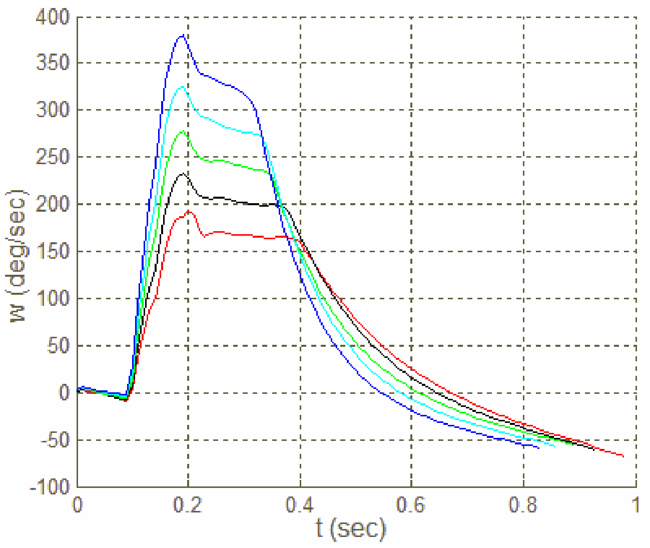

Fig 6. Angular velocities with the simple controller

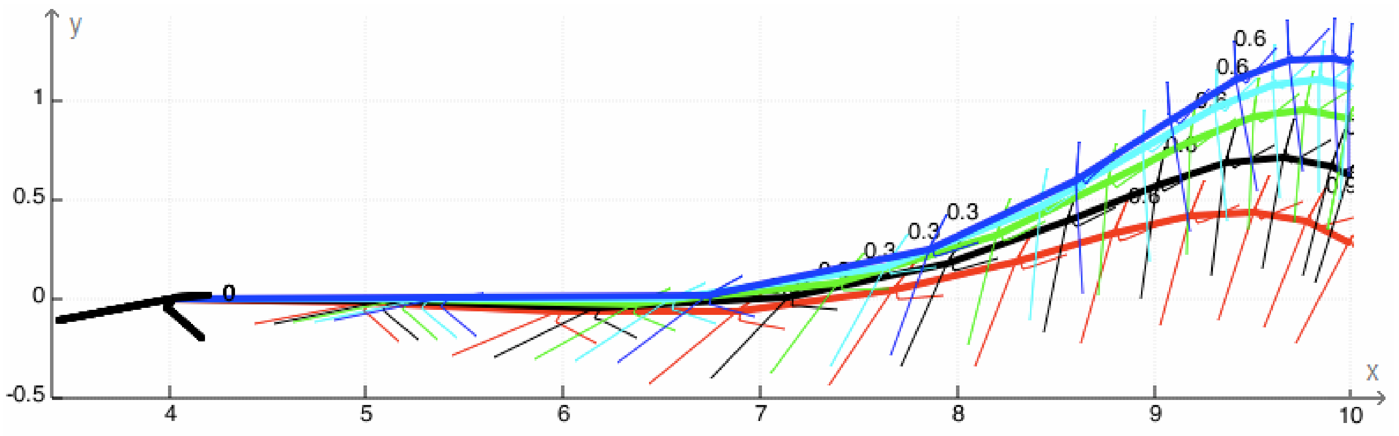

Fig 7. Simulated trajectories of the plane with the simple controller

Figure 7 shows the resulting trajectories generated with this controller. We can see that each landing is successful and quite similar to the trajectories created with the RL algorithm. The simple controller only requires a little amount of memory, uses sensors that are already onboard and has only a few parameters, allowing us to fine tune the trajectories using physical intuition.

Conclusion

In this paper, we presented a solution for the control of a wall landing airplane. Using a simulator, we used a value iteration algorithm to make an infinite horizon cost function converge in a coarsely discretized space. Implementation of several discretizations and cost function initializations allowed us to obtain excellent landings. One of them let us deduce a simple controller that achieves quality landings for different initial speeds in the simulator.

Although the simple control algorithm describe in this paper is working in simulation, two main issues still need to be investigated before implementing this controller on the airplane: the low frequency and discrete nature of the onboard controller (20Hz) and the low bandwidth of the elevator servo (10-15Hz). Even if both of these reveal to be significantly affecting the behavior of the controller, it is easy to imagine a control law that chooses a feedforward elevator displacement based on the observed $\Delta\omega$.

References

- Cory, R. and Tedrake, R., *Experiments in Fixed-Wing UAV Perching*, Proceedings of the AIAA Guidance, Navigation, and Control Conference, AIAA, 2008.

- Lussier Desbiens, A. and Cutkosky, M. R. *Landing and Perching on Vertical Surfaces with Microspines for Small Unmanned Air Vehicles (UAVs),* Journal of Intelligent and Robotic Systems, 22 Oct. 2009.

- Lussier Desbiens, A. and Asbeck, A. and Cutkosky, M. R., *Scansorial Landing and Perching*, Proc. 14th International Symposium on Robotics Research, September 2009, Lucerne, Switzerland.