Trajectory Optimization for Perching Quadrotor Vehicles

William Selby, Madalyn Berns

***Abstract*— Unmanned aerial vehicles (UAVs) have been carefully optimized for forward flight regimes over the past fifty years. While engineers learned to construct highly maneuverable craft to be flown by human operators, autonomous control of such highly underactuated systems has continued to be a difficult issue. New flight capabilities for aircraft have been enabled through recent advances in materials, actuators, and control architectures. Vertical perching, or the act of landing a craft on a wall or other vertical surface, is one of these newly enabled capabilities. Vertical perching would drastically increase the usable landing space of a craft, allowing it to conserve energy, collect sensor data, and potentially recharge even when there exists no suitable horizontal landing environment. For this study, we chose to use a quadrotor platform. Quadrotors are ideal for applications in which stability, maneuverability, and payload capacity are at a premium. We compare in simulation two trajectory optimization algorithms – Back-Propagation Through Time (BPTT) and Direct Transcription (DT) – to design a quadrotor control policy for vertical perching. To stabilize the system in real-time, we implement a LQR based trajectory stabilization policy to reject environmental disturbances. Our methods are successful in simulation and provide a foundation for future hardware implementation.**

I. INTRODUCTION

Advances in nonlinear control theory and hardware have made it possible to generate and autonomously control complicated trajectories in underactuated systems.

A. Aggressive Control of UAVs

Quadrotors are a prime candidate for implementing aggressive control maneuvers. Extremely stable in hover and horizontal maneuvers, they can also perform more complicated stunts [1]. However, the most aggressive maneuvers can only be performed successfully by humans, or through control policies learned by copying human control inputs [1]. In general, expanding the standard capabilities beyond horizontal flight and hover will allow quadrotors to more dynamically interact within a crowded and obstacle ridden operational theatre. Furthermore, exclusive autonomous control allows human operators to be focused on higher levels tasks rather than stabilizing the craft during complex motions.

Limitations of current MAVs	Hybrid platform advantages
Short mission life, especially if using electric power.	Long missions due to low energy consumption while perched. Possibility of solar charging.
Small MAVs have fast dynamics that complicate video and ranging surveillance.	Clinging MAV is a stable observation platform. It can also crawl to reorient as desired.
Fragile MAVs that land on the ground are vulnerable to accidental or malicious damage.	MAVs clinging high above the ground are out of harms way.
Ground may be cluttered with debris, making it hard to find space for landing and takeoff.	Walls remain relatively free of debris. Spines are not affected by film of dirt or moisture.
MAVs that land on horizontal surfaces are vulnerable to gusts of wind and inclement weather.	MAVs that cling to building walls can resist wind gusts. If perched under overhangs, they can ride out inclement weather.
MAVs are expending energy and making noise while moving.	Perched MAVs can be stationary and quiet, producing minimal detectable emissions.
Perching on a wire or pole: Detection of a wire or pole for autonomous perching is difficult. Wires are flexible and do not provide a stable observation platform.	Walls are easy to detect and provide a stable observation platform. They are also common in an urban environment.
Autonomous take-off usually requires a runway and consumes significant energy.	Jump-assisted take-off from a wall lets the plane reach airspeed quickly while clearing obstacles.

Fig. 1. Advantages of Perching UAVs [2]

B. Advantages of Perching

Vertical perching is the act of landing on a wall or other vertical surface. Perching provides many advantages for quadrotors on difficult missions as depicted in Figure 1. Landing on a wall allows the craft to conserve energy, continue sensing, and potentially recharge. This means that platforms can remain in theatre far longer than standard flight endurance. Perching also increases the number of viable landing regions. Regular horizontal landing regions may be debris-ridden and far apart, therefore it is imperative that craft have other landing options. Finally, vertical perching can keep a UAV safe (ex. from harsh winds), and quiet, able to continue sensing without detection.

Recently a number of scientists have begun to

Fig. 2. Flying Squirrel Landing on Wall [5]

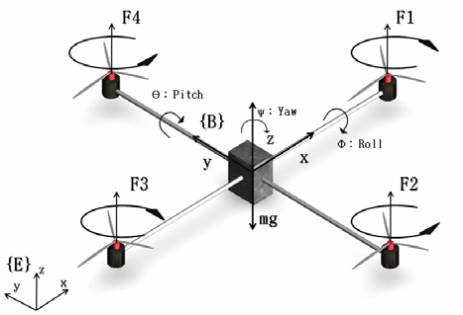

Fig. 3. Quadrotor Model [8]

work on the general perching problem. Tedrake and the Robot Locomotion Group at MIT have worked on perching both fixed and flap wing planes on poles and powerlines [3]. Cutkosky and the Biomimetics and Dextrous Manipulation Lab at Stanford have produced a vertical perching drone to land and grab onto walls with bio-inspired claws [2]. The Lab for Intelligent Machine Systems at Cornell have focused on vehicle morphing (using more complicated mechanical design) and trajectory optimization to solve the perching problem [4]. To the authors' knowledge, no one is currently working on perching quadrotors.

C. Connection to Animal Biomechanics

Biologists currently understand much about forward flight. However, less research has taken place on the transitions during perching and take-off. While, no animal illustrates the exact set of characteristics we would wish to replicate in a mechanical system, animal biomechanics does provides some interesting tidbits from a number of different animal systems that might provide clever solutions to the perching problem. Vertical perching in animals occurs in both flying and gliding creatures. Flying creatures like bats, birds, and insects perform different and potentially applicable maneuvers. Gliding creatures like flying squirrels, [5] as well as lizards and geckos [6], [7] also illustrate interesting perching behaviors.

II. Model

The model for the simulations are based on the Ascending Technologies Hummingbird Quadrotor platform. Unlike a typical helicopters which uses varying blade pitch angles to maneuver, a quadrotor uses differential thrust between the four motors to thrust, roll, pitch, and yaw. These commands also define the four inputs for our control system. The equations for the aerodynamic forces and control inputs are constructed using a basic nonlinear model of the quadrotor platform from [8]. A diagram of the quadrotor model is reproduced below in Figure 3. In the following equations, several assumptions are made. The body is assumed to be rigid and symmetrical and the center of mass and body fixed frame origin coincide. Aerodynamic effects such as blade flapping were ignored and the difference of gravity due to altitude or the spin of the Earth was neglected [9]. These assumptions and basic dynamics lead to the model shown below which was used for the analysis.

While this problem is essentially two-dimensional, a full three dimensional state matrix was chosen. The states include translational and rotational positions and velocities as defined in Figure model. The state equations are shown as Equation 1.

TABLE I

MODEL PARAMETER VALUES

Variable	Parameter	Value
m	Mass	0.5 kg
K_{fx}	Drag Coefficient (x)	0.01 kg/s
K_{fy}	Drag Coefficient (y)	0.01 kg/s
K_{fz}	Drag Coefficient (z)	0.01 kg/s
l	Arm Length	0.25 m
I_x	Moment of Inertia (x)	0.005 $kg \cdot m^2$
I_x	Moment of Inertia (y)	0.005 $kg \cdot m^2$
I_x	Moment of Inertia (y)	0.005 $kg \cdot m^2$
K_l	Lift Coefficient	0.05 $\frac{kg \cdot m}{s}$
K_d	Drag Coefficient	0.01 $\frac{kg \cdot m}{s}$
g	Gravity	9.8 m/s^2
R	Rotor Radius	0.1 m

$$
\begin{aligned}
\ddot{x} &= \frac{(cos\phi sin\theta cos\psi + sin\phi sin\psi)u_1 - K_{fx}\dot{x}}{m} \\
\ddot{y} &= \frac{(cos\phi sin\theta cos\psi - sin\phi cos\psi)u_1 - K_{fy}\dot{y}}{m} \\
\ddot{z} &= \frac{(cos\phi cos\theta)u_1 - K_{fz}\dot{z}}{m} - g \\
\ddot{\phi} &= \frac{l(u_2 - K_l\dot{\phi})}{I_x} \\
\ddot{\theta} &= \frac{l(u_3 - K_l\dot{\theta})}{I_y} \\
\ddot{\psi} &= \frac{(u_4 - K_d\dot{\psi})}{I_z}
\end{aligned}
\tag{1}
$$

where K_{fx}, K_{fy}, K_{fz} are the translational drag coefficients, l is the distance from the quadrotor center of mass to the center of the rotors, K_l is the lift coefficient, K_d is the rotational drag coefficient. I_x, I_y, and I_z are the moments of inertia (as given by the manufacturers). As drag is negligible at low speeds the coefficients were set to small values. The lift coefficient was arbitrarily determined. A precise measurement of this value is difficult because the plastic propellers (standard on this quadrotor) warp slightly with age. The values used are shown in Table I.

The control inputs to this system are defined in Equation 2 where u_1 is a thrust input, u_2 is the roll input, u_3 is the pitch input, and u_4 is the yaw input.

$$
\begin{aligned}
u_1 &= F_1 + F_2 + F_3 + F_4 \\
u_2 &= F_4 - F_2 \\
u_3 &= F_3 - F_1 \\
u_4 &= F_1 + F_3 - F_2 - F_4
\end{aligned}
\tag{2}
$$

Each motor produces a maximum of 0.35 kg thrust which converts to roughly 3.8 N of force. In addition, the motors are not back-drivable. This means the saturation limits for the thrust and yaw inputs are roughly 0 N - 13.7 N and $\pm$6.9 N for the roll and pitch inputs.

Finally, aerodynamic ground-effect was modeled for the quadrotor's interaction will the wall. As the quadrotor nears the wall, the general thrust increases as the air is pushed down through the rotors and reflected off the wall. However, once the quad reaches a certain proximity to the wall, the rotor-wash creates an area of low pressure between the quadrotor and wall which pulls the quadrotor towards the wall. This effect was modeled using Equation 3 from [10].

$$
\frac{T_{IGE}}{T_{OGE}} = \frac{1}{1 - \frac{R^2}{16z^2}} \tag{3}
$$

where T_{IGE} is the thrust in ground effect, T_{OGE} is the thrust out of ground effect, R is the rotor radius, and z is the distance of the quadrotor center of mass from the wall.

III. TRAJECTORY OPTIMIZATION ALGORITHMS

The goal of this study is to find a feasible control policy that would allow the quadrotor platform to rotate 90 degrees by the time it had reached a vertical surface. In order to calculate this desired trajectory, several algorithms are investigated. These algorithms use sequential quadratic programming (SQP) to calculate the control policy and perform gradient descent using a quadratic cost function. To perform the SQP calculations, we used the SNOPT software package. By altering the constraints as well as the cost function and terminal costs, several feasible trajectories can be found.

A. Back-Propagation Through Time

The first algorithm implemented was Backwards Propagation Through Time (BPTT). This is a shooting method which first integrates the equations of motion forward in time and then calculates the gradient of the cost function backwards in time. Given the cost function in Equation 4 and initial conditions $\mathbf{x}(0)$,

$$J(\mathbf{x}_0) = h(\mathbf{x}(t), \mathbf{x_d}) + \int_0^T g(\mathbf{x}(t), \mathbf{u}(t))\, dt, \quad \mathbf{x}(0) = \mathbf{x}_0 \tag{4}$$

integrate the equations of motion $\dot{\mathbf{x}} = f(\mathbf{x}, \pi_\alpha(\mathbf{x}, t))$ from time t to T.

$$-\dot{\mathbf{y}} = \mathbf{F}_x^T \mathbf{y} - \mathbf{G}_x^T \tag{5}$$

Next, integrate the adjoint equation (Equation 5) backward until time t = 0 where,

$$\mathbf{F_x}(t) = \frac{\partial \mathbf{f}}{\partial \mathbf{x(t)}} + \frac{\partial \mathbf{f}}{\partial \mathbf{u(t)}} \frac{\partial \pi_\alpha}{\partial \mathbf{x(t)}}$$
$$\mathbf{G_x}(t) = \frac{\partial \mathbf{g}}{\partial \mathbf{x(t)}} + \frac{\partial \mathbf{g}}{\partial \mathbf{u(t)}} \frac{\partial \pi_\alpha}{\partial \mathbf{x(t)}}$$

The gradients of the cost function are given by Equation 6

$$\frac{\partial J(\mathbf{x_0})}{\partial \alpha} = \int_0^T [\mathbf{G}_\alpha^T - \mathbf{F}_x^T \mathbf{y}] dt \tag{6}$$

where,

$$\mathbf{F}_\alpha(t) = \frac{\partial \mathbf{f}}{\partial \mathbf{u(t)}} \frac{\partial \pi_\alpha}{\partial \alpha}, \quad \mathbf{G}_\alpha(t) = \frac{\partial \mathbf{g}}{\partial \mathbf{u(t)}} \frac{\partial \pi_\alpha}{\partial \alpha}$$

Thus, BPTT returns the control tape

$$[\mathbf{u}[n], \mathbf{u}[n+1], \cdots \mathbf{u}[N]]$$

in discrete time where $dt = .02$ and $N = 2$ seconds over the quadratic cost:

$$g(\mathbf{x}[n], \mathbf{u}[n]) = \mathbf{u}[n]^T \mathbf{R} \mathbf{u}[n], \quad \mathbf{R} = \frac{1}{20}$$

with a terminal cost:

$$h(\mathbf{x}[N], \mathbf{x_d}) = \mathbf{Q_{end}}(\mathbf{x}[N] - \mathbf{x_d})$$

$\mathbf{Q}$, $\mathbf{R}$, and $\mathbf{Q_{end}}$ are positive diagonal matrices, where the entries $\mathbf{Q_{ii}}$ penalize the relative errors in state variable $\mathbf{x_i}$ compared to the desired state $\mathbf{x_d}$, and $\mathbf{R}$ penalizes actions in $\mathbf{u_i}$. $\mathbf{Q_{end}}$ in particular has several non-zero entries to penalize terminal position. The $x(T)$ and $z(T)$ positions were given costs of 100, pitch angle θ was given a penalty of 1000 and $\dot{\theta}$ was penalized 50.

$$\mathbf{x_d} = \begin{bmatrix} 2 & 0 & 1.5 & 0 & 0 & 0 & \frac{-\pi}{2} & 0 & 0 & 0 & 0 & 0 \end{bmatrix}^T$$

This method provided trajectories that minimized the cost function and arrived at the goal, yet these trajectories were not physically feasible (ie. the quadrotor traveled beyond the wall). Since the optimized tape only consists of only control inputs, it was not possible to directly constrain the states.

B. Direct Transcription

While BPTT didn't allow constraints to be placed on the states, Direct Transcription (DT) optimizes over a tape containing the control inputs as well as the state trajectory. Equation 7 shows the goal of DT

$$\min_{\forall n, x[n], u[n]} J = \sum_{n=1}^{N} g(\mathbf{x}[n], \mathbf{u}[n]) \tag{7}$$

where

$$\mathbf{x}[n+1] = \mathbf{f}(\mathbf{x}[n], \mathbf{u}[n])$$

and the returned tape is of the form

$$w = [\mathbf{x}[n], \mathbf{x}[n+1] \cdots \mathbf{x}[N], \mathbf{u}[n], \mathbf{u}[n+1], \cdots \mathbf{u}[N]]$$

The cost function is the same as BPTT.

DT uses the column vector of gradients, $\mathbf{G} = [\frac{\partial \mathbf{z}}{\partial \mathbf{w}}]$ where $\mathbf{z}$ is a column vector with the first element being the cost to go, J, followed by the discrete dynamics written as constraints where $\mathbf{z} = f(\mathbf{x}[n+1], \mathbf{x}[n], \mathbf{u}[n]) = 0$.

Using DT, it was possible to limit the x position to locations in front of the wall. This resolved the BPTT problem where the quadrotor traveled beyond the wall. One can also easily add constraints on additional state vectors to form the policy into a physically realizable trajectory. For example, penalizing the z velocity could be important when

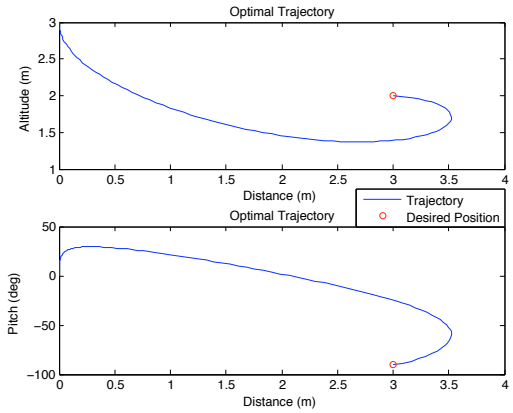

Fig. 4. BPTT General Trajectory

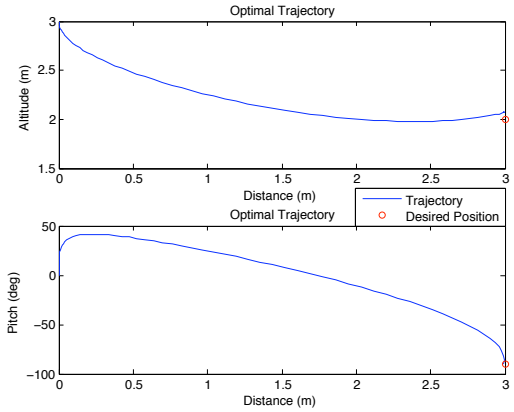

Fig. 5. BPTT Modified Trajectory

one considers that specific mechanisms to attach the quadrotor to the wall could perform differently based on the speed and direction the final approach.

C. Comparison of Methods

While the BPTT algorithm was easier to implement, the solutions were severely limited by the inability to add hard constraints on the states. For example, a straightforward implementation of the algorithm resulted in the trajectory shown in Figure 4.

From the figure, it is clear that this specific trajectory requires the quadrotor to maneuver beyond the wall at 3 m, something that is not physically possible. BPTT is subject to local minima in the gradient descent approach which may cause this result. However, it is possible to manipulate several variables to find more plausible trajectories. In Figure 4, the quad reaches a maximum x position of about 3.5 m at a pitch of around -50 deg. If the wall had been placed at 3.5 m and the control tape ended at this point, the quadrotor would be at the desired x position, but not the desired pitch. By decreasing the desired x position to account for the overshoot in this state as well as increasing the magnitude of the final pitch angle, a trajectory can be computed which has a feasible section. This is shown in FIgure 5.

The final desired x position was changed from 3 m to 2.8 m and the final desired pitch was modified from -90 degrees to -112.5 degrees. While executing the entire control tape will force the quadrotor to the modified non-desireable state, terminating the control tape early results in an x position of 2.99 meters with a pitch angle of -90.14 degrees. While these tricks manage to get a plausible trajectory, massaging the initial conditions to produce the desired policy is not a robust method of control. Other methods such as increasing the state penalty matrix as a function of distance away from the desired state were applied without any significant change to the general trajectory found by BPTT in Figure 5.

Next, DT was used to find a trajectory less subject to local minima and with the ability to directly penalize the states. The x state was constrained to eliminate all trajectories which contained an x position beyond the wall location. Using this DT implementation, the trajectory shown in Figure 6 was calculated.

This trajectory is physically feasible and could potentially be implemented in hardware. One possible limitation of this trajectory is the terminal z velocity. Depending on the type of mechanism used to attach the quadrotor to the wall, a high z velocity may not be beneficial. Instead, a trajectory which takes a more normal approach to the wall with a lower z velocity would be preferred. Constraining the z velocity state to -1.0 m/s, the trajectory shown in Figure 7 was found.

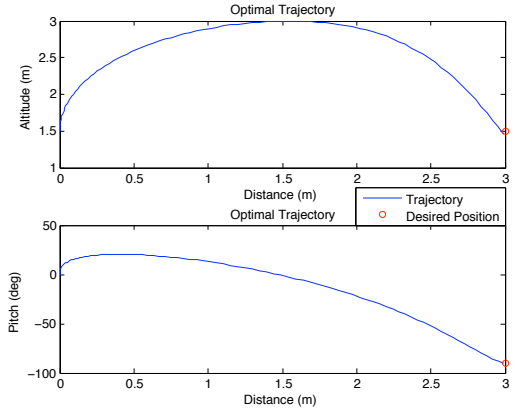

Fig. 6. DT General Trajcetory

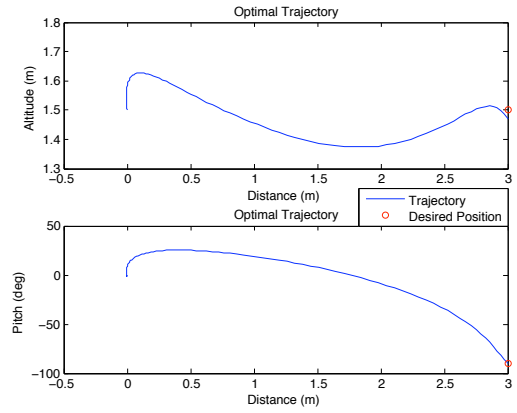

Fig. 7. DT Modified Trajectory

This trajectory takes a more direct path to the wall with a lower terminal z velocity. It is also possible to achieve a similar trajectory by penalizing the terminal z velocity in the terminal cost but this method seemed more sensitive to the penalty value. Constraining the state directly was a more reliable approach.

IV. Trajectory Stabilization

While BPTT and DT are useful tools to find possible trajectories in simulation, they return open loop control policies which loose their optimality in the face of noise and disturbances. LQR trajectory stabilization was implemented on the time-varying system to correct this problem. In order to linearize along the trajectory, a coordinate transformation was used. If $[\mathbf{x_0(t)}, \mathbf{u_0(t)}]$ is a feasible trajectory returned from BPTT or DT, then the new coordinate frame is defined by

$$\bar{\mathbf{x}}(t) = \mathbf{x}(t) - \mathbf{x}_0(t), \quad \bar{\mathbf{u}}(t) = \mathbf{u}(t) - \mathbf{u}_0(t)$$

.

This results in

$$\dot{\bar{\mathbf{x}}}(t) = \dot{\mathbf{x}}(t) - \dot{\mathbf{x}}_\mathbf{0}(t) = \dot{\mathbf{x}}(t) - \mathbf{f}(\mathbf{x}_0(t), \mathbf{u}_0(t))$$

which leads to the Linear Time-Varying (LTV) system:

$$\begin{aligned}\dot{\bar{\mathbf{x}}}(t) &= \frac{\partial \mathbf{f}(\mathbf{x}_0(t), \mathbf{u}_0(t))}{\partial \mathbf{x}}(\mathbf{x}(t) - \mathbf{x}_0(t)) \\ &\quad + \frac{\partial \mathbf{f}(\mathbf{x}_0(t), \mathbf{u}_0(t))}{\partial \mathbf{u}}(\mathbf{u} - \mathbf{u}_0(t)) \\ &= \mathbf{A}(t)\bar{\mathbf{x}}(t) + \mathbf{B}(t)\bar{\mathbf{u}}(t)\end{aligned}$$

Discretizing the equations with the same properties as the trajectory. Solving for the discrete time varying $\mathbf{A}[n]$ and $\mathbf{B}[n]$ results in

$$\begin{aligned}\mathbf{A}[n] &= \mathbf{I} + (\frac{\partial \mathbf{f}}{\partial \mathbf{x}} \mid_{\mathbf{x_0[n]}, \mathbf{u_0[n]}}) \, dt \\ \mathbf{B}[n] &= (\frac{\partial \mathbf{f}}{\partial \mathbf{u}} \mid_{\mathbf{x_0[n]}, \mathbf{u_0[n]}}) \, dt\end{aligned}$$

The cost is discretized as

$$\mathbf{Q} = \mathbf{Q}dt \quad \mathbf{R} = \mathbf{R}dt \quad \mathbf{Q_{end}} = \mathbf{Q_{end}}dt$$

and $\mathbf{S}[N] = \mathbf{Q_{end}}$.

The Discrete time Ricatti Equation, Equation 8, is solved recursively.

$$\begin{aligned}&\mathbf{S}[n-1] = \mathbf{A}^T[n-1](\mathbf{S}[n] - \mathbf{S}[n]\mathbf{B}[n-1] \\ &\cdot(\mathbf{B}^T[n-1]\mathbf{S}[n]\mathbf{A}[n-1]+\mathbf{R})^{-1}\mathbf{B}^T[n-1]\mathbf{S}[n])+\mathbf{Q}\end{aligned} \tag{8}$$

This results in a stabilizing control policy, $\mathbf{u}^*[n]$, where the gain matrix K is defined by

$$\begin{aligned}\mathbf{K} &= (\mathbf{B}^T[n]\mathbf{S}[n+1]\mathbf{B}[n]+\mathbf{R})^{-1}\mathbf{B}^T[n]\mathbf{S}[n+1]\mathbf{A}[n] \\ \mathbf{u}^*[n] &= \mathbf{u_0}[n] - \mathbf{K}\bar{\mathbf{x}}[n]\end{aligned}$$

A. *Closed-Loop System Evaluation*

In order to explore the robustness of the closed-loop controller, the quadrotor was disturbed with three different perturbations in simulation: initial condition uncertainty, uniform 'wind', and constant time delay. The nominal control tape is optimized from the original initial conditions.

Placing the quadrotor at a location different than the nominal initial condition, simulates a real world disturbance. Trajectory stabilization easily overcomes this perturbation as seen in Figure 8.

During a uniform 'wind', the plane is accelerated backwards by a previously unanticipated force. As shown in Figure 9, without stabilization the open-loop policy falls short of the goal. With the addition of the trajectory stabilization, the quadrotor reaches the desired end state with the wind disturbance. Random noise adds random position and velocity perturbations to the quadrotor throughout the trajectory which trajectory stabilization is able to account for as well.

Finally, simulated time delay was used to investigate the effects of lag in a hardware system which could be result from network latencies or insufficient processor power. If the optimal control tape $\mathbf{u_0}[n]$, a time delay T is added such that the implemented policy is $\mathbf{u_0}[n+T]$. In simulation, this causes the quad to perform the perching maneuver too late and therefore crashing into the wall at the wrong pitch angle. While LQR-based trajectory stabilization is sufficient to correct this problem for small, regular delays, Model-Predictive Control would be a better solution for a real-world hardware implementation.

As seen in Figures 8-9, the LQR-based trajectory stabilization algorithm alone successfully corrects the perturbed open-loop trajectories back to the goal by the final timestep. One drawback to trajectory stabilization is the lack of constraints on the new optimal control policy. Using the trajectory optimization algorithms, it was possible to limit the control policy to the physical limits of the motors on the quadrotor. While this is not possible

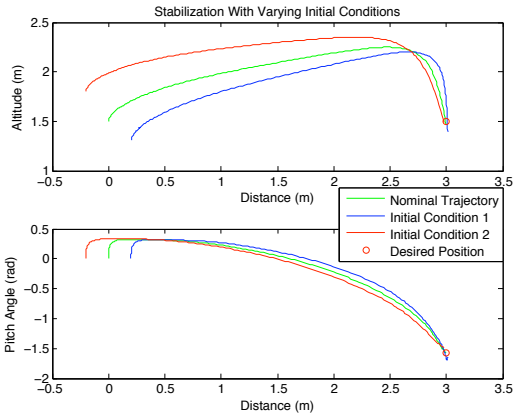

Fig. 8. Stabilization With Varying Initial Conditions

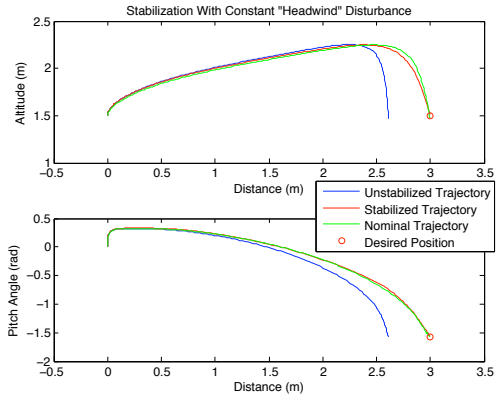

Fig. 9. Control Policy Response to "Headwind" Disturbance

using trajectory stabilization, the plot in Figure 10 shows that each of these closed-loop policies do not deviate significantly from the open-loop control and provide control values that remain within the bounds set for the physical actuators.

V. CONCLUSIONS AND FUTURE WORKS

A. *Conclusions*

This study investigated algorithms to determine feasible trajectories to place a quadrotor onto a vertical surface. Initially BPTT was investigated, but the inability to constrain the states explicitly

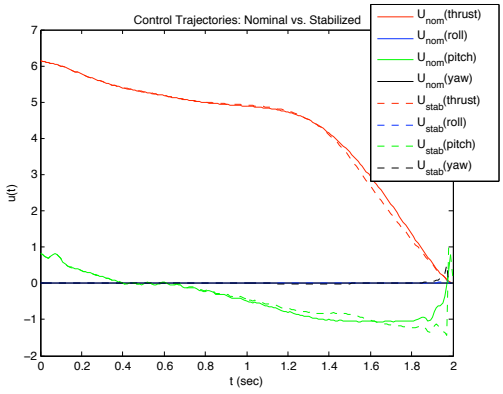

Fig. 10. Comparison of Nominal and Stabilized Control Policies

created physically impossible trajectories. DT was then used since it included the states vector as well as control inputs in the optimized tape. By limiting the x position, physically feasibly trajectories were calculated. Also, by limiting the z velocity, a trajectory which approached the wall at a perpendicular path was identified. Lastly, LQR based trajectory stabilization was used to create an optimal control policy robust to noise, varying initial conditions, and small time delays. This would be necessary for future hardware implementations.

B. Future Works

In order to move beyond simulation-based analysis, there are several changes that would need to be implemented to the current process. The current model uses dynamics which include a small angle approximation assumption which is inaccurate for aggressive maneuvers. Dynamics without these assumptions would make the algorithms more successful in the real-world. Secondly, we need to determine the proper mapping from our control tape output to quadrotor autopilot input values taking into account relevant motor curves and other nonlinear effects. In order to make our trajectory stabilization algorithms more successful, we would suggest implementing some form of Model-Predictive Control. Finally, some study may be necessary to determine the optimal physical mechanism for attaching to the wall. However, for initial experimenting and testing, velcro may ben an adequate substitute.

REFERENCES

[1] V. Gavrilets, E. Frazzoli, B. Mettler, M. Piedmonte, and E. Feron, "Aggressive Maneuvering of Small Autonomous Helicopters: A Human-Centered Approach," *The International Journal of Robotics Research*, vol. 20, no. 10, pp. 795–807, 2001.

[2] M. Cutkowsky and A. Desbiens, "Bio-inspired perching and crawling air vehicles," White paper, Stanford University, October 2008, available online (21 pages). [Online]. Available: http://bdml.stanford.edu/twiki/pub/Main/PerchingProject/SU-perching.pdf

[3] R. Cory and R. Tedrake, "Experiments in fixedwing uav perching," in *Proceedings of the AIAA Guidance, Navigation, and Control Conference*, 2008.

[4] A. Wickenheiser and E. Garcia, "Perching aerodynamics and trajectory optimization," Y. Matsuzaki, M. Ahmadian, and D. J. Leo, Eds., vol. 6525, no. 1. SPIE, 2007, p. 65250O. [Online]. Available: http://link.aip.org/link/?PSI/6525/65250O/1

[5] K. E. Paskins, A. Bowyer, W. M. Megill, and J. S. Scheibe, "Take-off and landing forces and the evolution of controlled gliding in northern flying squirrels Glaucomys sabrinus," *J Exp Biol*, vol. 210, no. 8, pp. 1413–1423, 2007. [Online]. Available: http://jeb.biologists.org/cgi/content/abstract/210/8/1413

[6] A. Jusufi, D. Goldman, and R. Revzen, S and. Full, "Active tails enhance arboreal acrobatics in geckos," *Proc. of the National Academy of Sciences of the United States of America*, vol. 105, no. 11, pp. 4075–4530, 2008.

[7] M. P. Murphy, B. Aksak, and M. Sitti, "Gecko-inspired directional and controllable adhesion," *Small*, vol. 5, no. 2, pp. 170–175, January 2009. [Online]. Available: http://dx.doi.org/10.1002/smll.200801161

[8] B. C. Min, C. H. Cho, K. M. Choi, and D. H. Kim, "Development of a micro quad-rotor uav for monitoring an indoor environment," in *Proceedings of the FIRA RoboWorld Congress 2009 on Advances in Robotics*. Berlin, Heidelberg: Springer-Verlag, 2009, pp. 262–271.

[9] H. Bouadi, M. Bouchoucha, and M. Tadjine, "Sliding mode control based on backstepping approach for an uav type-quadrotor," *International Journal of Applied Mathematics and Computer Sciences*, vol. 4, no. 1, pp. 12–17, 2008.

[10] K. P. Valavanis and K. P. Valavanis, *Advances in Unmanned Aerial Vehicles: State of the Art and the Road to Autonomy*. Springer Publishing Company, Incorporated, 2007.