AA203 Final Report: Quadcopter Optimal Path Planning in a Partially-Known Cluttered Environment

Manuel R. López Morales* Kyle Reinke †

Spring Quarter 2014

Abstract

The interest in this project arises from the desire to automate the delivery of light cargo within a city using quadcopters. In general, the layout of buildings and roads is known, but the obstacles that a quadcopter could encounter in a pre-defined route are not known. This limitation could be overcome by flying high, but this strategy will reduce the range of the trips the quadcopter can undertake because of battery capacity limitations. With this project we would like to become familiar with the challenges encountered in using low computational power to plan and track a path that minimizes a quadcopter's battery consumption.

The primary objective of the project is to reproduce the results shown in the paper *An Efficient Path Planning and Control Algorithm for RUAVs in Unknown and Cluttered Environments* by Yang et al[3]. In this paper, the authors apply a Rapidly-exploring Random Tree (RTT) algorithm to the problem of navigating a 3D space with obstacles. The simulated quadcopter becomes aware of obstacles as it navigates its environment. The authors use a Model Predictive Control (MPC) algorithm to follow a smoothed version of the piecewise path returned by the RTT algorithm.

We achieve similar results to those presented by Yang et al. In our simulations we are able to input obstacles in the form of surfaces, generate a path between arbitrary start and goal points without colliding against obstacles, and track the generated path with non-linear equations of motion.

1 Introduction

Unmanned Aerial Vehicles have seen an acceleration in interest for solutions to problems within environments that are cluttered with obstacles. In most cases not all obstacles are known at the initial path-planning stage, or that accurate representation of all known obstacles in memory is not feasible. This constraint gives rise to the need for an online path-planning methodology where the vehicle can sense its environment in and modify the flight path in real time.

For this project we considered the application of autonomous package delivery within an urban environment containing obstacles that are unknown before the mission begins. The vehicle begins at a specified start location, and must navigate the environment to reach the target delivery location without colliding with an obstacle. The overall strategy implemented to achieve this follows. At a specified frequency, the control loop first senses the environment to look for any new obstacles. If an obstacle is found that intersects the current planned path, a new path is computed. Once a path has been chosen, the control inputs to optimally track the path towards the goal are found. The inputs are then applied to the control until the the next period begins.

2 Obstacle Sensing

For the purposes of this short project, the sensor was assumed to be very simplistic. A radius of effective sensing from the vehicle's current location is defined in the simulation parameters. If any portion of an obstacle lies within this radius away from the vehicle's position, then the entire obstacle is sensed, and saved to the list of known obstacles. The sensor neglects any visual obstruction from other obstacles. For the simulations performed in this report, the sensing radius was assumed to be 10 meters.

*Ph.D. Candidate, Department of Aeronautics and Astronautics, Stanford University; mlopez14@stanford.edu
†M.S. Student, Department of Aeronautics and Astronautics, Stanford University; reinkek@stanford.edu

3 RRT

The problem of optimal path planning within a cluttered environment is far from trivial. The union of the numerous constraints imposed by the obstacles non-convex,. This makes finding a true optimal solution, especially in spaces of 3 or more dimensions, very computationally expensive. Instead, we applied a suboptimal path planning algorithm coded by Clifton, et. al. [2]

3.1 Path Finding

The path finding algorithm is based on the Rapidly-exploring Random Trees algorithm, which is a highly popular member of the random sampling class of motion planning algorithms. This particular implementation utilizes multiple randomly-exploring trees, one expanding from the start node, another expanding from the goal node, and potentially other seed nodes throughout the unoccupied space. As the trees expand, they look to connect a node from one tree to another without a collision with an obstacle. If such a connection is made, then the two trees are merged. Once all trees have been merged then a feasible path from the start to finish has been found, and the search algorithm exits.

3.2 Path Smoothing

Random sampling methods greatly improve the speed of computing a feasible solution, but often produce results that are far from optimal. A variety of smoothing techniques exist to shorten the path found by the RRT search. The RRT Matlab software also includes a smoothing stage. The smoothing algorithm takes advantage of the requirement that all obstacles are defined as planes in 3-D space. The point where the RRT crosses the extended planes is used to form an initial smoothing. Then those points are moved along the planes to the obstacle boundary if removing the point is not feasible. The smoothed curve is returned by the RRT planner as a series of line segments from the start to the goal. The final path that is returned is seen to be acceptably good for practical application on a consistent basis.

4 Equations of Motion

The non-linear dynamic model of a quadcopter used in our project is that derived by Bouabdallah [1]. The state of the system is

$$X = \left[\phi\ \dot{\phi}\ \theta\ \dot{\theta}\ \psi\ \dot{\psi}\ z\ \dot{z}\ x\ \dot{x}\ y\ \dot{y}\right]^T \tag{1}$$

and its dynamics are

$$\frac{dX}{dt} = \begin{pmatrix} \dot{\phi} \\ \dot{\theta}\dot{\psi}a_1 + \dot{\theta}a_2 U_4 + b_1 U_2 \\ \dot{\theta} \\ \dot{\phi}\dot{\psi}a_3 - \dot{\phi}a_4 U_4 + b_2 U_3 \\ \dot{\psi} \\ \dot{\theta}\dot{\phi}a_5 + b_3 U_4 \\ \dot{z} \\ g - (\cos\phi\cos\theta)\frac{1}{m}U_1 \\ \dot{x} \\ (\cos\phi\sin\theta\cos\psi + \sin\phi\sin\psi)\frac{1}{m}U_1 \\ \dot{y} \\ (\cos\phi\sin\theta\sin\psi - \sin\phi\cos\psi)\frac{1}{m}U_1 \end{pmatrix} \tag{2}$$

where $a_1, a_2, a_3, a_4, a_5, b_1, b_2, b_3, m$ are constants related to physical properties of the quadcopter like moments of inertia and mass; g is the acceleration due to gravity; U_1 is throttle control, U_2 is roll control, U_3 is pitch control, and U_4 is yaw control. The elements of the state vector are understood by observing Figure 1

4.1 Linearization

To design a controller for the quadcopter using tools for linear dynamics it is necessary to have an appropriate linearization of the equations of motion. We would like to find matrices A, B, and G such that

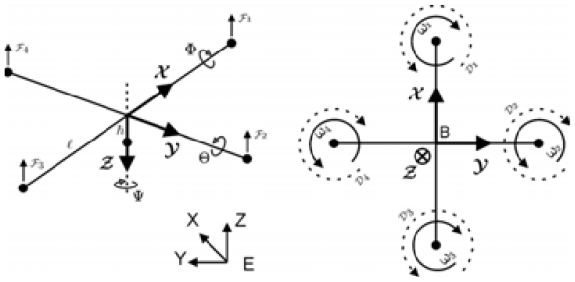

Figure 1: Quadcopter coordinate system [1]

$$\frac{dX}{dt} = AX + BU + G \tag{3}$$

where U is the control input vector $U = [U_1\ U_2\ U_3\ U_4]^T$.

By observing Equation (2), we note that attempting a crude linearization about any given state could yield equations in which yaw, pitch, and roll controls are decoupled from displacements in space. This would lead to the problem of having to have an attitude controller wrapped by a spacial displacement controller.

Given that in this project we aim to explore the ability of MPC to handle complex systems, we choose to linearize the equations so the non-linear coupling between control inputs and state is taken into account. To this end, we add the control input vector to our state vector, and control the system via a change in the control inputs. By doing this, the state vector becomes

$$X = \left[\phi\ \dot{\phi}\ \theta\ \dot{\theta}\ \psi\ \dot{\psi}\ z\ \dot{z}\ x\ \dot{x}\ y\ \dot{y}\ U_1\ U_2\ U_3\ U_4\right]^T \tag{4}$$

and the linearized equations of motion become

$$\frac{dX}{dt} = \begin{pmatrix} \hat{A} & \hat{B} \\ \mathbf{0} & \mathbf{0} \end{pmatrix} X + \begin{pmatrix} \hat{B} \\ \mathbf{1} \end{pmatrix} \delta U + \begin{pmatrix} \hat{G} \\ \mathbf{0} \end{pmatrix} \tag{5}$$

where $\hat{A}$ and $\hat{B}$ linearize the coupling between state variables and controls, and $\delta U = [\delta U_1\ \delta U_2\ \delta U_3\ \delta U_4]^T$ are the new control inputs. $\hat{G}$ contains constants coming from the coupled linearization.

To illustrate the procedure of finding the entries of matrices $\hat{A}, \hat{B}$, and $\hat{G}$, let us linearize the equation related to displacements in z (altitude). From the non-linear Equation in (2), we have

$$\ddot{z} = g - (\cos\phi\cos\theta)\frac{1}{m}U_1 \tag{6}$$

We can linearize $\cos\alpha$ about some angle α_0 by setting

$$\cos\alpha \approx \cos\alpha_0 + (\alpha - \alpha_0)(-\sin\alpha_0) \tag{7}$$

With this in mind, (6) can be linearized about point $(\phi_0, \theta_0, U_{10})$ as

$$\begin{aligned} \ddot{z} = g - \frac{1}{m}\Bigg(&\frac{1}{3}U_1\cos\phi\cos\theta \\ &+\frac{1}{3}U_1\cos\phi\cos\theta \\ &+\frac{1}{3}U_1\cos\phi\cos\theta\Bigg) \\ \approx g - \frac{1}{m}\Bigg(&\frac{1}{3}U_1\cos\phi_0\cos\theta_0 \\ &+\frac{1}{3}U_{10}\left[\cos\phi_0 + (\phi-\phi_0)(-\sin\phi_0)\right]\cos\theta_0 \\ &+\frac{1}{3}U_{10}\cos\phi_0\left[\cos\theta_0 + (\theta-\theta_0)(-\sin\theta_0)\right]\Bigg) \end{aligned} \tag{8}$$

Note that in Equation (8), each state variable ϕ, θ, U_1 appears once and multiplies constants. Entries in $\hat{B}$ correspond to constants multiplying the control state variables U, entries in $\hat{A}$ correspond to constants multiplying attitude and displacement state variables, and entries in $\hat{G}$ correspond to the constants not multiplying a variable.

5 MPC

5.1 Full Model with Linearized Equations

We can write Equation (5) as

$$\frac{dX}{dt} = \tilde{A}X + \tilde{B}\delta U + \tilde{G} \tag{9}$$

To put (9) in a form amenable to use in an MPC formulation, we approximate the change in time with a forward Euler formulation.

$$\frac{dX}{dt} \approx \frac{X(n+1) - X(n)}{\Delta t} = \tilde{A}X + \tilde{B}\delta U + \tilde{G} \tag{10}$$

where Δt is a time-step small enough so the linearized equations of motion are still a good approximation of the nonlinear equations.

$$X(n+1) = [\mathbf{1} + \Delta t\tilde{A}]X(n) + [\Delta t\tilde{B}]\delta U + [\Delta t\tilde{G}] \tag{11}$$

In order to test the accuracy of the linearization and to select a time-step Δt and a look-ahead number of steps N, we assessed how closely the linearized plant followed the behavior of the non-linear equations of motion. A value of $\Delta t = 0.01s$ and $N = 60$ yielded satisfactory results. Figure (2) shows that the linearized plant followed the general trend of the non-linear plant (quadcopter). The linearization happened at time $t = 0$ and the plant was advected in time by using Equation (11) for N steps. At intervals of $0.05s$, a random input value δU was entered in order to produce a response in the system.

5.2 Simplified Model

For the full simulation, a time-step of 0.01s was considered infeasible given the computation time required for the MPC algorithm. An alternate system was considered, which assumes that a low-level stability controller on-board the UAV gives a closed-loop plant that behaves approximately as a point-mass with a thrust-vector input. The force vector can be interpreted as an attitude command input to the stability controller, and a scaling of the motor thrusts such that the desired thrust magnitude is maintained.

The complete MPC model framework for this simplified model used in the simulations shown in the attached videos is given mathematically below:

$$\text{minimize } C_p \sum_{i=1}^{N+1} \left\| \left(\frac{(p_s - p_g)^T (p_s - p_g)}{\|p_s - p_g\|_2^2} - I \right) (x^{(i)} - p_g) \right\|_2 + C_g \sum_{i=1}^{N+1} \|x^{(i)} - p_g\|_2 + \sum_{i=1}^{N} \|R^{\frac{1}{2}} u^{(i)}\|_2 \tag{12}$$

$$\text{subject to } \begin{bmatrix} x^{(1)} \\ \dot{x}^{(1)} \end{bmatrix} = \begin{bmatrix} x_1 \\ \dot{x}_1 \end{bmatrix} \tag{13}$$

$$u^{(0)} = u_0 \tag{14}$$

$$\begin{bmatrix} x^{(i+1)} \\ \dot{x}^{(i+1)} \end{bmatrix} = \begin{bmatrix} I & \Delta t I \\ 0 & I \end{bmatrix} \begin{bmatrix} x^{(i)} \\ \dot{x}^{(i)} \end{bmatrix} + \frac{\Delta t}{m} \begin{bmatrix} 0 \\ I \end{bmatrix} u^{(i)} + \Delta t \begin{bmatrix} 0 \\ g \end{bmatrix} \tag{15}$$

$$\|u^{(i)}\|_2 \leq u_{\max} \tag{16}$$

$$\sqrt{(u_1^{(i)})^2 + (u_1^{(i)})^2} \leq a_{\text{attitude}} u_3^{(i)} \tag{17}$$

$$\sqrt{(u_1^{(i)} - u_1^{(i-1)})^2 + (u_1^{(i)} - u_1^{(i-1)})^2} \leq a_{\text{rotation}} u_3^{(i-1)} \tag{18}$$

The discretized linear system model of the alternate system used in the MPC is of the same form as (11) with different coefficient matrices as shown in the MPC constraint (15). A constraint is placed on the MPC optimization due to the finite tracking ability of the low-level controller. The UAV is restricted to a finite change in attitude command per time as

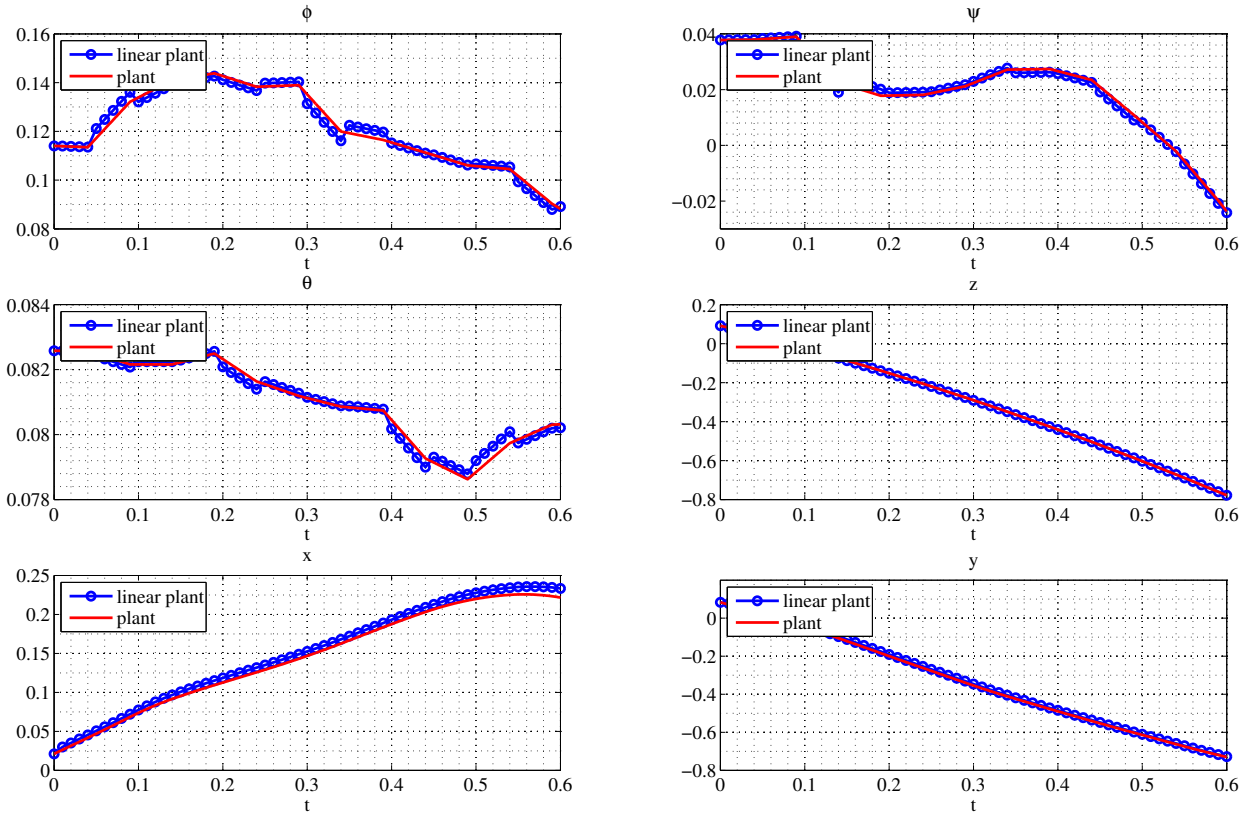

Figure 2: Behavior of linearized equations of motion with respect to non-linear equations

expressed in equation (18). The UAV attitude is restricted such that the angle between body-fixed and earth-fixed z-axes cannot exceed 45 degrees by constraint equation (17). The maximum combined thrust from the 4 motors is restricted to be less than 100 Newtons (16).

The objective function (12) for the MPC minimizes a combination of the distance of the current location to the line containing the current line segment, the euclidean distance from the current location to the line segment's goal node, and the size of the thrust-vector.

6 Results

The very limited time-step of $\Delta t = 0.01s$ did not allow us to complete a full simulation of the quadcopter's tracking the path provided by RRT using the full non-linear plant. However, to demonstrate that the MPC does provide satisfactory controls for these very non-linear equations, we used MPC to do a simple reference tracking. We start at the point $(x, y, z) = (0, 0, 0)$ and ask the quadcopter to move to point $(1, 1, -1)$. Note that a value of $z = -1$ is equivalent to climbing 1 meter. The results of this simulation are presented in Figure (3). We notice that the linearization of the equations provide a very satisfactory result. The weight on the reference error is 10 and on the changes in control δU is 0.005. The change in control was limited to a magnitude of $1,000$. Because of time-constraints, we were not able to test a more sophisticated cost function.

What is remarkable in these results is that it was indeed possible to let MPC select direct control inputs to the quadcopter's motors and it was not necessary to impose any constraints on the quadcopter's attitude. The proper linearization of the equations was sufficient to allow the MPC algorithm to exploit the coupling between the attitude and the displacements to find a useful solution.

In order to show that it is possible to get satisfactory results from coupling the MPC algorithm with RRT, we performed simulations on a simplified model of a quadcopter as described in 5.2. To effectively demonstrate the value of MPC versus an open-loop optimization of inputs, additional components were added to the simulation dynamics. Random Gaussian

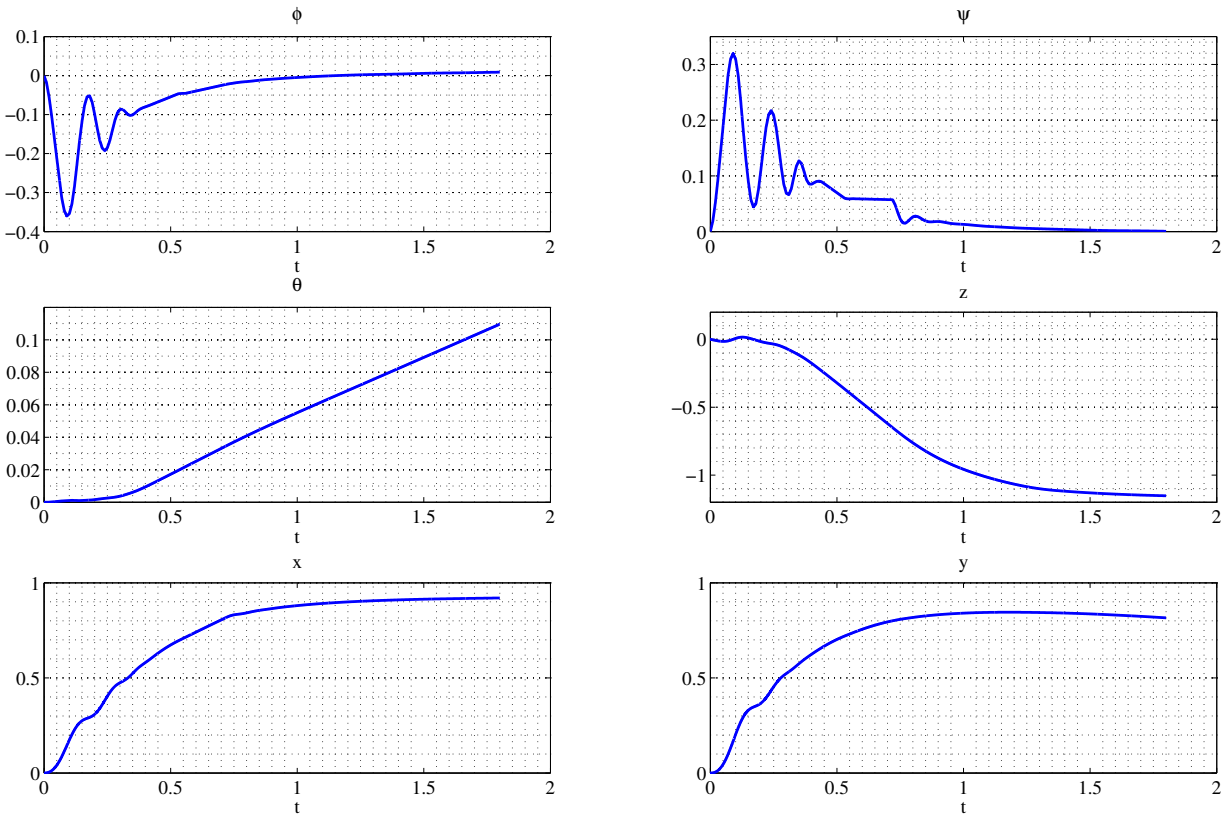

Figure 3: Attitude and displacement history of a quadcopter moving from $(x, y, z) = (0, 0, 0)$ to $(1, 1, -1)$ using MPC

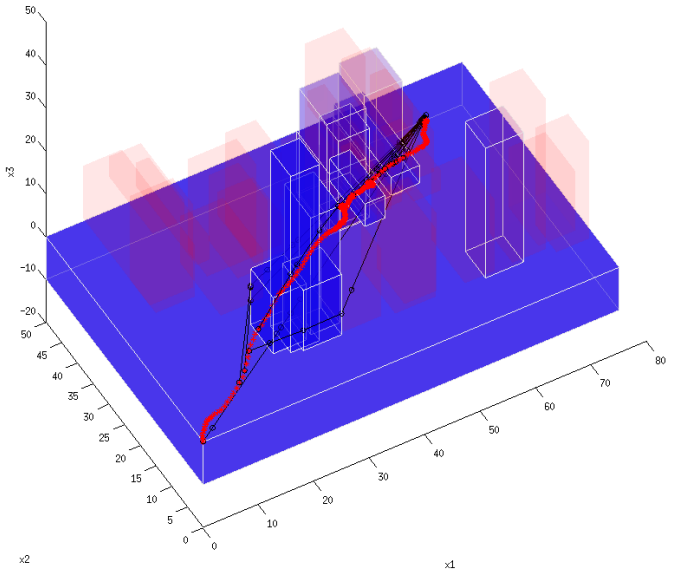

Figure 4: Picture of a completed simulation run with simplified dynamics. Disturbance forces, sensor uncertainty, and second order drag force are included in this simulation. MPC provides the controls for tracking the line. In blue are the observed obstacles, in red are the unobserved but existing obstacles. The path to track is in black. The quadcopter's center of mass is in red.

noise was added into the position input of the MPC to simulate GPS measurement uncertainty. A randomly changing disturbance was added to the input vector to simulate windy conditions. Also, Nonlinear aerodynamic drag was added to the equations of motion. Without a closed-loop control, these effects could not be compensated for.

Videos of the complete simulations for the simplified model are attached. In these videos, the UAV follows the path provided by the RRT until it encounters a new obstacle, at which time a new path is generated and tracked. A picture of one such simulations is in Figure (4).

The RRT algorithm worked flawlessly in always finding a path from the arbitrary origin to the arbitrary goal. In the attached videos, unobserved obstacles are in red, observed obstacles are in blue. The darker the blue, the longer time the quadcopter has been aware about the obstacle.

7 Conclusion

The encouraging results shown in this project neglect many important factors that must be accounted for before the method will be suitable for use on a physical vehicle. The first, and biggest, factor is the time delay incurred by computation time of the RRT and MPC algorithms. The code will need to be optimized, parallelized, and ported to a faster language such as C. Some time delay will be inevitable, but when accounted for with a sufficiently small delay and accurate dynamic model, this factor will become manageable.

Nevertheless, we showed that combining RRT with MPC is a very powerful control strategy. RRT was always able to find a feasible path and MPC was able to control the quadcopter without the need of much tuning.

The importance of the way the equations of motion are linearized is evident in our results. The fact that we were able to let the coupling of the attitude and displacements become evident in the linearization is what allowed us to use such a generic MPC for a plant as non-linear as a quadcopter. We were unable to find in the existing literature linearized equations of motion of a quadcopter in the form presented here. This theoretical development could be exploited to

develop very generic LQR or LQG algorithms for quadcopters; there would be no need to tune attitude control separately from displacement control.

References

[1] Samir Bouabdallah. Design and control of quadrotors with application to autonomous flying. *Lausanne Polytechnic University*, 2007.

[2] Matthew Clifton, Gavin Paul, Ngai Kwok, Dikai Liu, and D-L Wang. Evaluating performance of multiple rrts. In *Mechtronic and Embedded Systems and Applications, 2008. MESA 2008. IEEE/ASME International Conference on*, pages 564–569. IEEE, 2008.

[3] Kwangjin Yang, Seng Keat Gan, and Salah Sukkarieh. An efficient path planning and control algorithm for RUAVs in unknown and cluttered environments. *Journal of Intelligent and Robotic Systems*, 57(1-4):101–122, 2010.