MOTION PLANNING UNDER UNCERTAINTY: APPLICATION TO AN UNMANNED HELICOPTER

A Thesis

by

JOSHUA DANIEL DAVIS

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

August 2006

Major Subject: Aerospace Engineering

MOTION PLANNING UNDER UNCERTAINTY: APPLICATION TO AN UNMANNED HELICOPTER

A Thesis

by

JOSHUA DANIEL DAVIS

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Approved by:

Chair of Committee,	Suman Chakravorty
Committee Members,	John L. Junkins
	Nancy Amato
	Aniruddha Datta
Head of Department,	Helen Reed

August 2006

Major Subject: Aerospace Engineering

ABSTRACT

Motion Planning Under Uncertainty: Application to an Unmanned Helicopter.

(August 2006)

Joshua Daniel Davis, B.S. Auburn University

Chair of Advisory Committee: Dr. Suman Chakravorty

A methodology is presented in this work for intelligent motion planning in an uncertain environment using a non-local sensor, like a radar sensor, that allows the sensing of the environment non-locally. This methodology is applied to an unmanned helicopter navigating a cluttered urban environment. It is shown that the problem of motion planning in a uncertain environment, under certain assumptions, can be posed as the adaptive optimal control of an uncertain Markov Decision Process, characterized by a known, control dependent system, and an unknown, control independent environment. The strategy for motion planning then reduces to computing the control policy based on the current estimate of the environment, also known as the "certainty equivalence principle" in the adaptive control literature. The methodology allows the inclusion of a non-local sensor into the problem formulation, which significantly accelerates the convergence of the estimation and planning algorithms. Further, the motion planning and estimation problems possess special structure which can be exploited to reduce the computational burden of the associated algorithms significately. As a result of the methodology developed for motion planning in this thesis, an unmanned helicopter is able to navigate through a partially known model of the Texas A&M campus.

ACKNOWLEDGMENTS

This thesis is the result of the contributions and efforts of many individuals. First, I would like to thank Dr. Suman Chakravorty for his time as well as input, which has made me more mature in my understanding of motion planning for autonomous systems and has made the development of the high-level motion planner possible. I would also like to thank Colonel Steven Suddarth from the Air Force Research Laboratory for providing a Bergen Observer Helicopter as well as an Automated Flight Control System from Rotomotion, LLC. He also spent many hours helping me understand the flight controller hardware and the programming necessary to interface with its software. I am grateful to Dennis D'Annunzio from Rotomotion for providing meaningful help and support. David Lund and Reza Langari invested considerable time and resources in the helicopter to ensure it flew successfully and were critical to this project's success. I would also like to thank Chris Mentzer, a fellow graduate student, who did more than his fair share of assembling the helicopter and preparing it for flight testing.

I am appreciative to Dr. John Junkins, who played an instrumental role in my acceptance to Texas A&M and providing my first year of funding. It was a pleasure to sit under his teaching as well as gain wisdom from his words of advice on technical and non-technical issues. I would like to thank Dr. Helen Reed for her financial support during my second year at Texas A&M. It has been a great joy to work with her as a teaching assistant and I am thankful for all that I have been able to learn about satellites under her guidance. I would like to thank Dr. Aniruddha Datta and Dr. Nancy Amato

for their support as members of my committee. Thank you to many of my fellow graduate students such as Jaime Ramirez, Jeff Morris, David Oertli, Sowan Suman, Sunny Jain, Sakthivel Kasinathan, and Shashank Shukla. They have been exceptional friends and made my graduate school experience more enjoyable.

I am thankful to Jesus Christ, who gives purpose and meaning to life. He has provided encouragement and strength to me through many passages in the Bible. I am also thankful to others, such as my friends in the Navigators Christian student organization for their encouragement and friendship, which has enhanced my experience while at Texas A&M. Finally, I would like to thank my family who has been an amazing support and has played a crucial role in my success in this endeavor.

TABLE OF CONTENTS

Page

LIST OF FIGURES

FIGURE Page

NOMENCLATURE

Symbol	Description
$\overline{c}$	= average transition cost
D	= number of possible environment states at any system state s
$E_{\mu}(\ )$	= expectation operator with respect to policy μ
F^t	= history of process until time t
$J^*(\ )$	= optimal cost-to-go
$\overline{J}(\ )$	= average cost-to-go
M	= number of control actions possible
N	= number of system states
$p(\ )$	= true environmental uncertainty
$P_t(\ )$	= vector of the true environment probabilities at time t
$q(\ )$	= environment state
$\hat{q}(\ )$	= non-locally observed environment state
$\overline{q}(\ )$	= possible environment state at r
$\mathbb{R}^D$	= real space of D dimensions
Q	= set of all possible environment states at each system state
s	= system state
s'	= state from which observation occurs
s_t	= system state at time t
S	= set of all possible system states

Symbol	Description
t	= time
T	= set of all times at which the control policy is updated during the path planning
$\bar{T}$	= average dynamic programming operator
u	= control action
$u^*(\ )$	= the optimal control
U	= set of all possible control actions
$\bar{V}$	= space of all probability vectors in $\mathbb{R}^D$
x	= x axis position coordinate
y	= y axis position coordinate
z	= z axis position coordinate
α	= tuning parameter for distance to the end
β	= infinite horizon discount factor
η	= transition cost to visit a state with no obstacle is
$\lambda(\ )$	= factor to make the sensor model stochastic
$\pi(F(s))$	= fraction of time that system is in the footprint of set *F(s)*
$\pi(\hat{q}(s))$	= probability for observing noise corrupted environment state $\hat{q}(s)$
$\pi(s')$	= fraction of time that the system is at s'
$\Pi_t(s)$	= vector includes the π_t values for each environment s
μ^*	= optimal control policy
$\mu_k(\ )$	= control update at time instant t_k
$\gamma_t^{ij}(\ )$	= a D x D matrix which is a combination of all the sensor models used and their associated number of measurements
$\Gamma_t(\ )$	= sensor model of uncertainty
ζ	= cost of hitting an obstacle

Acronyms	Description
AFCS	Automated Flight Control System
API	Application Programming Interface
ASIC	Application Specific Integrated Circuit
cc	cubic centimeters
DOF	Degree of Freedom
DP	Dynamic Programming
FLTK	Fast Light ToolKit
GB	Gigabyte
GHz	Gigahertz
GPS	Global Positioning System
IMU	Inertial Measurement Unit
IP	Internet Protocol
MDP	Markov Decision Process
MB	Megabyte
OpenGL	Open Graphics Library
PCM	Pulse Code Modulated
PID	Proportional Integral Derivative
POMDP	Partially Observed Markov Decision Process
RC	Remote Control
UAV	Unmanned Aerial Vehicle

Acronyms	Description
UGV	Unmanned Ground Vehicle
VORAD	Vehicle Onboard RADar
WAAS	Wide Area Augmentation System

1. INTRODUCTION

In this thesis, a methodology for the motion planning of an autonomous agent in an uncertain environment is proposed and applied to an unmanned helicopter navigating a cluttered urban environment. The optimal path for the unmanned helicopter is planned using a priori knowledge about the environment and the sensor data received, as the helicopters navigates through obstacles in the environment. The motion planner involves a high-level planner which plans against the uncertainty in the environment and issues its commands in a series of waypoints for a lower-level controller to track. A commercially available lower-level controller from Rotomotion LLC, called the Automated Flight Control System (AFCS), was interfaced with the high-level motion planner so that the motion planning algorithm could be implemented on a six degree of freedom flight simulator.

The high-level controller plans the motion of the agent in an uncertain environment using a radar sensor. The methodology is applicable for any non-local sensor, i.e., a sensor that allows sensing of environment states surrounding the current system state. The state space of any motion planning problem can be expressed as the ordered pair *(s,q(s))* were *s* represents the system state and *q(s)* represents the state of the environment at the state *s*. For example, in the case of an unmanned helicopter exploring an urban environment, *s* corresponds to the *(x,y,z)* coordinates of the helicopter

---

This thesis follows the style of *Journal of Guidance, Control, and Dynamics.*

And *q(s)* corresponds to the presence or absence of an obstacle at the point *(x,y,z)*. The goal of the motion planning strategy is to use all available information about the environment, until the current time instant, in order to plan the "best possible" path. It is known that the planning problem can be modeled as a Markov Decision Process (MDP), characterized by a known, control dependent exploration system and unknown, uncontrollable environment.[1,2] Our formulation allows the integration of a radar or a similar non-local sensor into the planning methodology. The planning and estimation problems, as formulated in this thesis, have special structure which can be exploited to significantly reduce the dimensionality of the associated algorithms.

There has been substantial research in the adaptive control of controlled Markov Chains, or Markov Decision Processes, in the past two decades. In indirect adaptive control, the transition probabilities of the underlying Markov Chain are estimated and the control is applied based on the most recent estimate of the transition probabilities.[3,4,5] This is known as the so-called "certainty equivalence principle". The "direct" approach to stochastic adaptive control falls under the category of "reinforcement learning" methodologies wherein the optimal control is calculated directly with resorting to estimating the transition probabilities of the underlying Markov Chain.[6,7,8] Underlying all these methods is Bellman's "principle of optimality" or Dynamic Programming, which is a methodology for sequential decision-making under uncertainty.[9,10] In this work, it is shown that the motion planning problem can be reduced to the adaptive optimal control of a Markov Decision Process and thus the above methodologies can be applied to the same. The indirect approach to adaptive control is adopted since mapping

the environment is of interest too. The general representation for an infinite horizon control problem based upon the principle of optimality can be represented by:

$$\pi^*(s_0) = \underset{\pi=(u_1,u_2,\ldots)}{\arg\min}\, E\left(\sum_{t=1}^{\infty} \beta^k c\left(s_{t+1} / s_t, u_t\right) / s_0\right). \quad (1)$$

[11]

$\pi^*(s) = (u_1, u_2, \ldots)$ represents the optimal control policy that starts at the state s_0, while $c\left(s_{t+1} / s_t, u_t\right)$ is the cost to transition from state s_t with control u_t to s_{t+1}. The term β represents the discount factor for the infinite horizon problem and discounts the terms as the states become farther out on the horizon. This formulation allows a feedback control to be implemented for a system, using dynamic programming.

As additional states are added to a dynamic programming problem there is a geometric growth in computation, which is considerably more attractive than a direct enumeration method for control determination, which would give an exponential growth in computation.[12] The reason that direct enumeration requires such a large amount of computation, is that it determines all possible control sequences and compares them to determine an optimal path. Even though dynamic programming is more efficient computationally than direct enumeration, the greatest challenge associated with dynamic programming is "the curse of dimensionality". This means that as the number of dimensions or states increase, the computational requirements rapidly increase and can cause the solution to become computationally unfeasible.

Motion planning considers how the state space (configuration space) is represented. The dimension of the configuration space depends upon the degree of freedom of the robot being used in the motion planning.[13] Another consideration is if the system is

modeled as continuous or discrete. If the system is modeled as discrete, the grid size (number of states) will greatly effect the computation time.

Various approaches have been developed for collision-free motion planning of unmanned systems in known environments.[14] In the past decade, there has been an increasing interest in the case when the environment in which unmanned system is operating is partially or completely unknown. The uncertainty in the environment is treated as a deterministic worst case[15,16] or on a probabilistic average case basis.[17] In "probabilistic robotics", there has been substantial research in the localization of a mobile robot while simultaneously mapping the environment.[18,19,20,21,22] In Ref. 14, a game-theoretic framework is proposed for robotic motion planning. The authors resort to Bellman's principle of optimality[9] in order to tackle the motion-planning problem. In Ref. 14, a Bayesian adaptive control[4] approach is formulated which suffers from the issue of dimensionality and may not be suitable for high dimensional environments.[23] A non-Bayesian adaptive control framework is adopted in our approach.

There has been a blending of previously different areas of study: planning, control theory, and artificial intelligence for motion planning. In the past, planning has focused more on planning the trajectory of a vehicle, while control theory has focused on the response of differential equations to control inputs.[24] The area of artificial intelligence in the past tented to focus on problem solving in a discrete state space.[25,26] The development of algorithms for autonomous systems to navigate through obstacle fields has caused the differences between planning algorithms and control theory to become less distinguished.

Many of motion planning techniques that have been implemented are essentially graph search methods, such as breath first search, depth first search, and Dijkstra's algorithm. Breadth first search considers all possible paths that are equal distance from the starting point. Although it is not the most computationally efficient it guarantees the optimal path is determined.[27] Depth first search considers the cost from a starting point to the goal state along a feasible path and may later consider alternate paths.[28] Dijkstra's algorithm is a single source shortest path algorithm and was developed as a special form of dynamic programming. Dijkstra's algorithm includes several heuristics to draw the vehicle toward the goal state and help eliminate unnecessary computations by removing unnecessary state analysis.[29]

One of the most well know graph search methods for motion planning is A*, pronounced ("ay-star"), and was created as a method for determining an optimal path or trajectory by including heuristics. It determines an admissible, "optimistic", solution. A* is effective for a priori planning of a static environment but requires complete recalculation if the environment changes.[30] A* and Dijkstra's algorithm are essentially the same except for the function which is used to sort the vector recording the cost of feasible paths.[23] A development in A* research led to a class of algorithms called Focused Dynamic A* (D*),[31] which includes heuristics as well as incremental search techniques so that a complete recalculation of the costs is not necessary for a dynamically changing environment. D* has been implemented for various motion planning problems, such as indoor robots, outdoor UGV (unmanned ground vehicles) in the DARPA Unmanned Ground Vehicle Program,[32] urban robots, and even the Mars

rover.[33] A newer version of D* called D* Lite has been developed that is at least as efficient and often more efficient than D*. D* Lite is also more intuitive to understand than D* and has been rigorously analyzed mathematically.[34]

Logic based methods also exist as a possible solution to motion planning and can be implemented similar to graph search methods, but take a somewhat different approach to motion planning.[35,36] Logic methods often focus on partial plans and sub-goals. There are various implementations to logic based planning, for example, planning graphs can be analyzed by layer construction, or the planning problem can be tackled using a Boolean approach.

Probabilistic roadmaps (PRM) are a recent development in motion planning and are an efficient method for determining an optimal path. They are essentially a sample-based approach to navigation of an environment.[37] The roadmap is a graph of randomly generated collision-free paths, which are connected by a simple fast planning method.[38] The advantage of using a roadmap is the efficiency with which the nodes (waypoints) are connected in the configuration space. If part of the roadmap is not used then those computations are wasted. However there are probabilistic roadmap variants, which minimize unnecessary computations. The samples can be chosen randomly for the state space (configuration space) to obtain meaningful information in developing a model of the environment. In the study of roadmaps important information includes the denseness of the samples chosen as well as the method of choosing the samples. Roadmaps require preprocessing before the vehicle begins navigation and essentially act as a network of multiple pairs of initial-goal points. In the implementation of the algorithm the roadmap

construction phase is used to generate the nodes and connect them, while the query phase is used to evaluate which path is optimal. The "probabilistic" part of the name comes from the fact that the method performs sampling in a probabilistic fashion. However, probabilistic roadmaps do not perform control in a probabilistic fashion.

The original contributions of the current work are as follows. We propose a hierarchical motion planner wherein, the problem of "intelligent motion planning" for the high-level planner is reduced to the adaptive optimal control of an uncertain Markov Decision Process, characterized by a known, control dependent system and an unknown, control independent environment; and a low-level controller is then used to track the commands from the high-level planner. The methodology allows for the inclusion of non-local sensors, which significantly reduce the computational burden of the estimation and planning algorithm in an uncertain environment. The motion planning methodology is applied to the problem of a UAV helicopter navigating the Texas A&M campus and is tested on a six degree of freedom flight simulator of the helicopter.

The rest of the thesis is organized as follows. Section 2 details the structure of the motion planner. Section 3 contains the formulation of the motion planning problem as a Markov decision problem. In section 4, we present the results of implementing the planning methodology on a UAV helicopter navigating the Texas A&M campus by testing the algorithms on a six DOF flight simulator.

2. CONTROLLER STRUCTURE

The motion planning algorithm is implemented in a hierarchical fashion wherein a high-level motion planner determines the optimal waypoints for the low-level controller to track during its traverse to the goal state. The planner takes into consideration the shortest distance to the goal state, local and non-local environment sensing, as well as the current state of the helicopter when it makes decisions. The high-level planner minimizes the distance traveled by choosing waypoints between the initial and goal state while avoiding obstacles during the flight. The high-level controller determines each desired waypoint and issues it to the low-level controller. The low-level controller receives the waypoint command and issues the necessary commands to the flight surfaces. Once the waypoint is achieved the high-level planner receives new sensor information about non-local states and determines the next waypoint. This process continues until the goal state is achieved. Figure 1 below outlines the basic motion planning architecture.

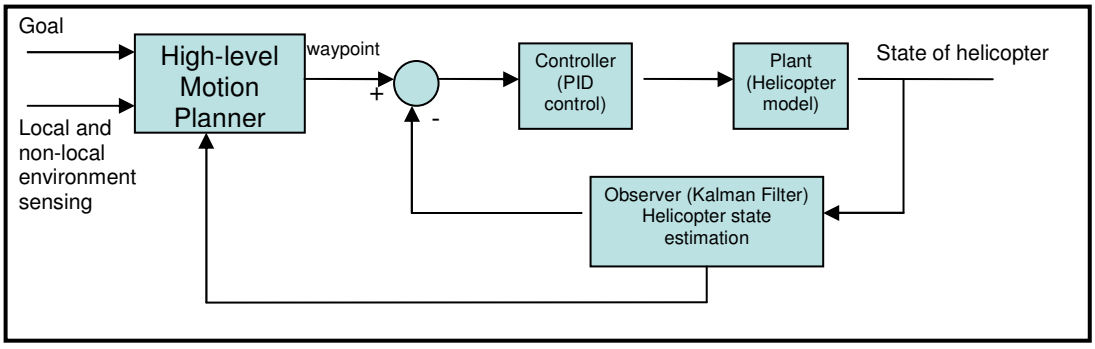

Fig. 1 Architecture for the High-Level Motion Planner and Low-Level Controller

The low-level flight controller used in the simulations was the Automated Flight Control System (AFCS), which was developed by Rotomotion LLC and is capable of executing waypoint commands given to the helicopter. The AFCS has a series of Proportional-Integral-Derivative (PID) control loops that stabilize the attitude, position, and velocity of a remote control helicopter.[39,40] The control loops use an observer (state estimator), which is implemented using a Kalman Filter. As the helicopter flies, the Kalman Filter also gives state feedback to the high-level motion planner so that the high-level planner knows that a waypoint has been achieved. It is assumed that the low-level controller has sufficient control over the helicopter so that it will not deviate from the grid and hit obstacles. The motion planner is thus in the form of a composite feedback control wherein the high-level planner plans against the uncertainty in the environment on a longer time/length scale while the lower-level controller is robust to uncertainties at shorter time/length scaling, and also for the dynamic uncertainties in the system model.

3. PLANNING UNDER UNCERTAINTY USING NON-LOCAL SENSING

First, we recount some results that will be required for the motion planning.[1,2]

A. Preliminaries

Let the state of the exploration system be denoted by s, $s \in S$. Denote the state of the environment at the system state s by $q(s)$. For example:

1) In the case of robotic exploration of unknown terrain, s corresponds to the (x,y) coordinates of the robot and $q(s)$ corresponds to the height of the terrain $z(x,y)$ at the point (x,y).
2) In the case of a UAV navigating enemy territory while avoiding radar detection, the s variable corresponds to the position (x,y,z) of the UAV while $q(s)$ corresponds to the binary valued variable indicating the presence or lack of radar coverage at the point (x,y,z).

From hereon assume that the system state is sensed perfectly and only the environment is sensed imperfectly. Let the number of system states be N and let the number of possible environment states, at any system state s, be D and denote this set by Q. Denote the local state of the exploration system by the ordered pair $(s, q(s))$. Let the set of control actions be denoted by U and let the total number of control actions possible be denoted by M. Denote any particular control action by u. The following Markovian assumption is made about the system. Let

$F^t = \{(s_0, q_0(s_0)), u_0, ..., (s_{t-1}, q_{t-1}(s_{t-1})), u_{t-1}\}$ represent the history of the process until time *t*.

A **3.1** T*he current system state, s_t, is dependent only on the system state and control input at the previous time instant, i.e.,*

$$p(s_t / F^t) = p(s_t / s_{t-1}, u_{t-1}). \tag{2}$$

A **3.2** *The environment process is "incoherent", i.e., the environment process is spatially uncorrelated and temporally stationary. In other words, if $\{q_t(s), s \in S\}$ denotes the environment process, $q_t(s)$ is a stationary process for all $s \in S$. Moreover, $q_t(s)$ is independent of $q_\tau(s')$ whenever $s \neq s'$, for all t, τ. Note that $q_t(s)$ is a random variable and the above assumption is used in a probabilistic sense.*

Deterministic environments (like an unstructured terrain) automatically satisfy the above assumptions.

Proposition 1 *Under assumptions A3.1,A3.2, the following holds:*

$$p((s_t, q_t(s_t)) / F^t) = p(s_t / (s_{t-1}, u_{t-1}))\, p(q_t(s_t)) \tag{3}$$

The transition probabilities $p(s_t / (s_{t-1}, u_{t-1}))$ quantify the control uncertainties inherent in the system and are assumed to be known beforehand. The environmental uncertainty *p(q(s))* is unknown and successive estimates are made of this uncertainty as the planning proceeds to completion. Motion planning may be framed as an infinite horizon discounted stochastic optimization problem, i.e., given the initial state $(s_0, q_0(s_0))$, the optimal control policy $\mu^*(s_0, q_0(s_0)) = \{u_1, u_2, ...\}$ is defined by:

$$\mu^*\left(s_0, q_0\left(s_0\right)\right) = \arg\min_\mu E_\mu\left(\sum_{t=1}^{\infty}\beta^t c\left(\left(s_t, q_t\left(s_t\right)\right),\left(s_{t-1,} q_{t-1}\left(s_{t-1}\right)\right), u_{t-1}\right) / \left(s_0, q_0\left(s_0\right)\right)\right) \quad (4)$$

where $c\left(\left(s_t, q_t\left(s_t\right)\right),\left(s_{t-1,} q_{t-1}\left(s_{t-1}\right)\right), u_{t-1}\right)$ is a positive pre-defined cost that the system incurs in making the transition from state $\left(s_{t-1,} q_{t-1}\left(s_{t-1}\right)\right)$ to $\left(s_{t,} q_t\left(s_t\right)\right)$ under the control action u_{t-1}, $E_\mu(.)$ denotes the expectation operator with respect to the policy μ, and $\beta < 1$ is a given discount factor.

The following environment sensing model is adopted:

3) At every instant *t*, the system (UAV) at state s_t, can observe the environment state $q_t(s)$, (i.e., the current environment state at the state *s*), if $s_t \in F\left(s\right) \subseteq S$, where *F(s)* is assumed to be known beforehand. The set *F(s)* constitutes a "footprint" of the sensor system.

4) Associated with every observation-vantage point pair, $(q(s), s'),\ s' \in F(s)$, (i.e., we are observing the environment at *s*, *q(s)*, from the state *s'*), there exists a known measurement error model, $p\left(\hat{q}\left(s\right) / q\left(s\right), s'\right), \hat{q}\left(s\right), q\left(s\right) \in Q$, and $s,\ s' \in S$, i.e., the probability that $\hat{q}(s)$ is observed when the environment is actually at the state *q(s)*, for an observation made from system state *s*. Such a model maybe deduced from sensor calibrations.

The estimation and control schemes that are used to tackle the motion planning problem are discussed next.

B. Estimation

Consider the following relationship:

$$\pi(\hat{q}(s)) = \frac{1}{\pi\left(F\left(s\right)\right)} \sum_{s' \in F(s), q(s)} p\left(\hat{q}(s) / q\left(s\right), s'\right) p\left(q\left(s\right)\right) \pi\left(s'\right) \tag{5}$$

where

1) $\pi\left(\hat{q}\left(s\right)\right)$ denotes the probability of observing the noise corrupted environment state $\hat{q}(s)$ during the course of the exploration, i.e., the fraction of the time that the environment at state *s* is observed to be at $\hat{q}(s)$ during the course of the exploration.
2) *p(q(s))* denotes the true probability that the environment state is *q(s)* at the state *s*.
3) $\pi(s')$ denotes the fraction of the time that the system is at state *s'* and $\pi(F(s))$ represents fraction of time that the system spends in the footprint set *F(s)*.

The above equation states that the frequency of observing a particular value of the environment during the course of the exploration process is related to the actual probability of the environment taking that value, the noise model and the frequency of visiting the states of the system. Since the noise model is known, and the values of $\pi\left(\hat{q}\left(s\right)\right)$ and $\pi\left(s\right)$ can be estimated during the course of the exploration using the Monte-Carlo method, it is possible to obtain the true environment probabilities using equation 5. Mathematically:

$$\pi_t(\hat{q}_n(s)) := \frac{1}{t} \sum_{1}^{t} 1(\hat{q}_n(s) = \hat{q}(s)) \tag{6}$$

$$\pi_t(s) := \frac{1}{t} \sum_{n=1}^{t} 1(s_n = s) \tag{7}$$

where *1(A)* denotes the indicator function of the event *A*. Then, the true probabilities of the environment process *p(q(s))* can be obtained recursively as:

$$P_t(s) := \arg\min_{P \in \bar{V}} \left\| \Pi_t(s) - \Gamma_t(s) P \right\|^2 \tag{8}$$

where

$$P_t(s) = \left[p_t(q_1(s)), ..., p_t(q_D(s)) \right]' \tag{9}$$

$$\Pi_t(s) = \left[\pi_t(\hat{q}_1(s)), ... \pi_t(\hat{q}_D(s)) \right]' \tag{10}$$

$$\Gamma_t(s) = \lambda(s) \left[\gamma_t^{ij}(s) \right] \tag{11}$$

$$\gamma_t^{ij}(s) = \sum_{s' \in F(s)} p(q_i(s) / q_j(s), s') \pi_t(s') \tag{12}$$

||.|| denotes the Euclidean norm in $\mathbb{R}^D$, and $\bar{V}$ represents the space of all probability vectors in $\mathbb{R}^D$. Thus, keeping account of the probabilities, $\pi(s)$ and $\pi(\hat{q}(s))$, the true probabilities of the environment process can be recovered asymptotically. The term $\lambda(s)$ is equal to $1/\pi\left(F(s)\right)$ and is included to make each row of $\Gamma_t(s)$ stochastic.

C. Control

Consider the stochastic optimal control problem posed in equation 4. Using the Bellman principle of optimality,[6] it can be shown that the optimal policy is stationary, i.e., the optimal control is independent of time, and that the optimal control at the state *(s,q(s))*, $\mu^*(s, q(s))$, is given by the following equation:

$$u^*(s, q(s)) = \arg\min_u \sum_{(r, \bar{q}(r))} p(r / s, u) p(\bar{q}(r)) [c((r, \bar{q}(r), (s, q(s)), u) + \beta J^*(r, \bar{q}(r))], \tag{13}$$

where $J^*(r,\overline{q}(r))$ is the optimal cost-to-go from the state $(r,\overline{q}(r))$. Moreover, J^* satisfies the following fixed point equation:

$$J^*(s,q(s)) = \min_u \sum_{(r,\overline{q}(r))} p(r/s,u)\, p(\overline{q}(r))[c((r,\overline{q}(r),(s,q(s)),u) + \beta J^*(r,\overline{q}(r))] \qquad (14)$$

The problem of path planning is one of adaptive control of an uncertain Markov Decision Process, (since the probabilities of the environment process are not known). However, the system state and the local environment are assumed to be sensed perfectly and thus the problem is not a Partially Observed Markov Decision Process (POMDP).[11] In such a scenario, the strategy of adaptive control is to use the policy that is optimal with respect to the current estimate of the system, since it corresponds to the current knowledge of the system that is being controlled and is referred to as the "certainty equivalence principle" in adaptive control.[3]

Let $T = \{t_1, t_2, ..., t_k, ...\}$ denote the set of all times at which the control policy is updated during the path planning. Let the updated control policy at time instant t_k be denoted by $\mu_k(s,q(s))$. Let $p_t(q(s))$ denote the estimated environmental uncertainty at the time t, obtained from the estimation equation 8. Then, the control update at time $t_k \in T, \mu_k(.)$, using the principle of optimality and the "certainty equivalence principle", is given by

$$\mu_k\left(s,q(s)\right) = \arg\min_u \sum_{(r,\overline{q}(r))} p\left(r/s,u\right) p_{t_k}\left(\overline{q}\left(r\right)\right)\left[c\left(\left(r,\overline{q}\left(r\right)\right),\left(s,q\left(s\right)\right),u\right) + \beta J_k(r,\overline{q}(r))\right] \qquad (15)$$

where

$$J_k\left(s,q(s)\right)=\min_u \sum_{(r,\bar{q}(r))} p\left(r/s,u\right) p_{t_k}\left(\bar{q}\left(r\right)\right)\left[c\left(\left(r,\bar{q}\left(r\right)\right),\left(s,q\left(s\right)\right),u\right)+\beta J_k(r,\bar{q}(r))\right].$$

(16)

Next, a few of the salient properties of the control problem posed are listed. The following result establishes the convergence of the cost-to-go functions to the optimal cost-to-go function when the estimates of the environmental process converge.

Proposition 2 *Under assumptions A2.1-A2.2, the cost-to-go functions* $J_t(s,q(s)) \to J^*(s,q(S))$ *as* $t \to \infty$, *if* $p_t(q(s)) \to p(q(s))$, *for all* $s \in S$, $q(s) \in Q$.

The optimality equations can be simplified based on the special structure of the path planning problem due to the incoherent environment assumption, which allows us to reduce the dimensionality of the dynamic programming problem. Consider the optimality equation:

$$J(s,q(s))=\min_u \sum_{(r,\bar{q}(r))} p(r/s,u)\,p(\bar{q}(r))[c((r,\bar{q}(r),(s,q(s)),u)+\beta J(r,\bar{q}(r))] \quad (17)$$

Let

$$\bar{c}\left(s,u,q\left(s\right)\right)=\sum_{(r,q'(r))} p\left(r/s,u\right) p\left(\bar{q}\left(r\right)\right)c\left(\left(r,\bar{q}\left(r\right)\right),\left(s,q\left(s\right)\right),u\right). \quad (18)$$

Noting that

$$p(r,\bar{q}(r))/(s,q(s)),u)=p(r/s,u)\,p(\bar{q}(r)), \quad (19)$$

it follows that

$$J\left(s,q(s)\right)=\min_u\left[\bar{c}\left(s,u,q\left(s\right)\right)+\beta\sum_r p\left(r/s,u\right)\bar{J}(r)\right], \quad (20)$$

where

$$\bar{J}(s) = \sum_{q(s)} p\left(q(s)\right) J\left(s, q(s)\right). \tag{21}$$

Noting that

$$u^*\left(s, q(s)\right) = \arg\min_u \left[\bar{c}\left(s, u, q(s)\right) + \beta \sum_r p\left(r / s, u\right) \bar{J}\left(r\right) \right], \tag{22}$$

it can be concluded that an average value of the cost-to-go function $J(s, q(s))$, at the system state s, namely $\bar{J}(s) = \sum_{q(s)} p(q(s)) J(s, q(s))$, is required in order to be able to evaluate the optimal control at any state $(s, q(s))$. This allows us to carry an "average" feedback control. However, there still remains the problem of estimating the average cost-to-go vector $\bar{J}(s)$. In order to answer this question, note that

$$\bar{J}\left(s\right) = \sum_{q(s)} p\left(q(s)\right) \min_u \left[\bar{c}\left(s, u, q(s)\right) + \beta \sum_r p\left(r / s, u\right) \bar{J}(r) \right]. \tag{23}$$

Hence, $\bar{J}$ is the fixed point of the "average" dynamic programming operator $\bar{T} : \mathbb{R}^N \to \mathbb{R}^N$, defined by the following equation:

$$\bar{T}\ \bar{J}\left(s\right) = \sum_{q(s)} p\left(q(s)\right) \min_u \left[\bar{c}\left(s, u, q(s)\right) + \beta \sum_r p\left(r / s, u\right) \bar{J}(r) \right], \forall s\,. \tag{24}$$

The following proposition states that the "average DP operator", $\bar{T}$, is a contraction mapping that maps $R^n \to R^n$ and thus, the optimal average cost-to-go vector, which is unique fixed point, can be obtained using successive approximations.[41] This allows us to significantly reduce the computational burden of the planning algorithm.

Proposition 3 *The average DP operator, $\bar{T}$, is a contraction mapping in $\mathbb{R}^N$ under the ∞ norm.*

4. TESTING ON AN UNMANNNED HELICOPTER

The methodology presented thus far in this thesis was used for motion planning of an unmanned helicopter in an uncertain environment. The motion planning algorithms were implemented in a six DOF simulation of a UAV helicopter navigating through an urban environment model of the Texas A&M campus. The results show that the planning methodology proposed maybe suitable for autonomous navigation in a cluttered urban environment.

A. Developing an urban environment

An environment model of the inner part of the Texas A&M campus was developed in MATLAB, using maps as well as aerial photos of the campus from Google Earth. Once a three dimensional campus model was developed, it was overlaid with the aerial images for visualization purposes.

The system state, *s*, represents the *(x,y)* grid point coordinate of the vehicle and the environment state, $q(s)$, represents the presence of an obstacle or otherwise at that particular grid point. A 750x 550 meter area on the campus of Texas A&M was discretized into 75 x 55 grid of points (*i* rows, *j* columns) so that the number of possible system states *N* was equal to 4125. The environment state at any grid point has 2 possible values, obstacle or no obstacle. Figure 2 shows that at any given location the helicopter had four possible high-level control actions: Go (Forward/Back/Right/Left) to the adjacent grid point.

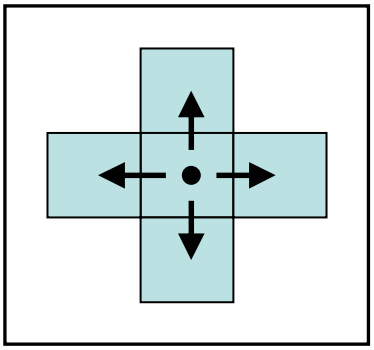

Fig. 2 Possible High-Level Control Actions

B. Non-local state estimation using a radar

A non-local sensor for motion planning would ideally have unlimited range, a 360 degree field of view, and no uncertainty. However, a perfect level of accuracy is not possible in practical applications, and there are multiple sensors available that can be used in motion planning. Researchers have used millimeter wave radar, LIDAR (LIght Detection and Ranging), infrared sensors, and ultrasonic sensors. LIDAR sensors as well as millimeter wave radars have been implemented on full size helicopters to warn pilots of obstacles. A millimeter wave radar (24.7 GHz) called the Eaton VORAD (Vehicle On-board RADar) was chosen as the non-local sensor to be modeled in this research. The VORAD has an operating range of 1 meter to 107 meters (3 feet -350 feet), with an uncertainty of 5%, and a field of view of 12 degrees, with an uncertainty of 0.2 degrees. The radar antenna, onboard processor, and batteries require a helicopter with a lift capability of 6-7lbs. The radar software was developed to track up to seven obstacles every 65ms (15Hz), by reporting azimuth to each obstacle, range to each obstacle, and gain of the return from each obstacle. The simulation treats all obstacles as having the same radar reflectivity characteristics. Two changes were made to the

sensing model to accommodate the simulation: first, the field of view was changed to 90 degrees and second, radar tracking was set to nine obstacles.

C. Interfacing with a low-level flight controller

A low-level flight controller known as the Automated Flight Control System (AFCS), developed by Rotomotion Inc. for unmanned helicopters, was used to track the high-level flight control commands. The AFCS uses two extended Kalman filters for state estimation: the first is a 7 state Kalman filter, 4 states for quaternion and 3 states for gyro bias, and the second is 6 state Kalman filter, 3 states for North East Down *(NED)* position and 3 states for orthogonal velocity components *(UVW)*. The flight controller implements nine Proportional Integral Derivative (PID) loops- three for position *(x,y,z)*, three for velocity $(\dot{x}, \dot{y}, \dot{z})$, and three for attitude (ψ, θ, ϕ).

The AFCS position sensor utilizes the Wide Area Augmentation System (WAAS) Global Positioning System (GPS) and gives position measurements to within 3 meters accuracy 95% of the time.

D. Motion planning in the flight simulator

D.1 Navigation and exploration

The high-level motion planner calculates an initial cost-to-go map $\overline{J}(s)$, which is an array of *i* rows and *j* columns, based upon the a priori environment data available. Before the high-level motion planner implements a control, it senses and updates probabilities of the environment state at 8 of the adjacent grid-points. If the helicopter is commanded in a direction that has not been sensed by the radar sensor, the helicopter

changes directions and performs new measurements of the non-local environment states since the sensors in this case have only a 90 degree field of view. Then the motion planner recalculates the control and the command is issued to the low-level flight controller for execution.

The operation of the high-level planner involves four different tuning parameters. The cost associated with hitting an obstacle is $\zeta = 100$, while the transition cost to visit a state with no obstacle is $\eta = 1$. Various tuning values were used for ζ and η. The only importance of the numbers for ζ and η is their magnitude relative to each other. These values are important because the cost-to-go to the goal state includes the transition cost along with the cost-to-go from a future state and the distance of a future state from the goal state. An infinite horizon discount factor of $\beta = .99$ was used to ensure that states in the immediate future have a greater effect on the control than states farther out on the horizon. For lower β values the vehicle is more likely to be trapped once the vehicle enters a boxed in area. Also a tuning parameter of $\alpha = 6$ was included in the simulations to represent a weighting value for distance to the goal. For large α values the vehicle is drawn to the final destination and hits obstacles, while the value of $\alpha = 6$ draws the vehicle to the final destination but does not hit obstacles. It is necessary to tune the parameters because the path planning problem is posed as an infinite horizon problem. The structure of the control is given by:

$$u^*\left(s, q(s)\right) = \arg\min_u \left[\bar{c}\left(s, u, q(s)\right) + \beta \sum_r p\left(r / s, u\right) \bar{J}\left(r\right) \right]$$

where the cost-to-go is:

$$\bar{J}(s) = \sum_{q(s)} p\left(q(s)\right) \min_{u} \left[\bar{c}\left(s, u, q(s)\right) + \beta \sum_{r} p\left(r / s, u\right) \bar{J}(r) + \alpha d(r) \right].$$

The $\bar{c}\left(s, u, q(s)\right)$ term represents the average transition cost from a state, *s*, to an adjacent state, *r*, given that the control is *u*. The term $\alpha d(r)$ represents the tuning parameter α multiplied by the distance of the future state, *r,* to the goal state.

Figures 3 through 15 included an a priori environment model accuracy of 90%. This number quantifies the reliability of the a priori environment model. Various levels of a priori environment model accuracy were used in the initial testing of the algorithm. If the a priori environment model accuracy was treated as $p_0\left(q(s)\right) = 0.5$, then the a priori model did not have a meaningful contribution to the motion planning (since a 50% reliability is as good as flipping a fair coin). As the a priori accuracy of the model increased from 0.5 to 0.9 the initial motion plan became more reliable and less likely to hit obstacles.

The sensor model was generated by simulating sensor data using a Monte Carlo approach for development of the sensor model. Each sensor measurement was taken as the true measured value and combined with a Gaussian noise based upon the distance to the obstacle. The simulations using the Eaton VORAD model had an accuracy of $p\left(q(s') / q'(s'), s\right) = 0.95$ for the non-local state sensing; i.e. when $(s' \neq s)$.

The simulations were also performed with various different uncertainty models so that the effects of using non-local sensors, with different accuracy levels, on the performance of the algorithm could be quantified. For example, in figures 5, 6, and 7 the non-

directional sensor model has an accuracy of $p\left(q\left(s'\right)/q'\left(s'\right),s\right)=0.96$ for the local sensing, an accuracy of $p\left(q\left(s'\right)/q'\left(s'\right),s\right)=0.93$ at a distance of 10 meters, and an accuracy of $p\left(q\left(s'\right)/q'\left(s'\right),s\right)=0.91$ at a distance of 14 meters. This model was developed by assuming non-local measurements at 1 meter have an accuracy of $p\left(q\left(s'\right)/q'\left(s'\right),s\right)=0.96$, while measurements at 100 meters have an accuracy of $p\left(q\left(s'\right)/q'\left(s'\right),s\right)=0.57$. The accuracy of the radar for a given range, was found by linearly interpolating between the accuracy of the sensor at 1 meter and 100 meters. The figures 8 through 15 use a directional sensor model with an approximate accuracy of $p\left(q\left(s'\right)/q'\left(s'\right),s\right)=0.99$. For a sensor uncertainty of $p\left(q\left(s'\right)/q'\left(s'\right),s\right)=0.5$, the vehicle performed as if no sensor data was available. For sensor uncertainties between $p\left(q\left(s'\right)/q'\left(s'\right),s\right)=0.7$ and $p\left(q\left(s'\right)/q'\left(s'\right),s\right)=0.8$, the performance increased significantly. For uncertainties in the range of $p\left(q\left(s'\right)/q'\left(s'\right),s\right)=0.9$ to $p\left(q\left(s'\right)/q'\left(s'\right),s\right)$=0.99, the obstacle avoidance performance remained approximately the same.

The motion planning software was tested multiple times before executing the waypoints in the flight simulator. Figures 3 and 4 have been included below to show a path that the high-level flight controller generated to navigate through campus while using a priori map data as well as radar sensor information. The plots have been overlaid with aerial photos to display the simulated vehicle trajectory through the Texas A&M campus.

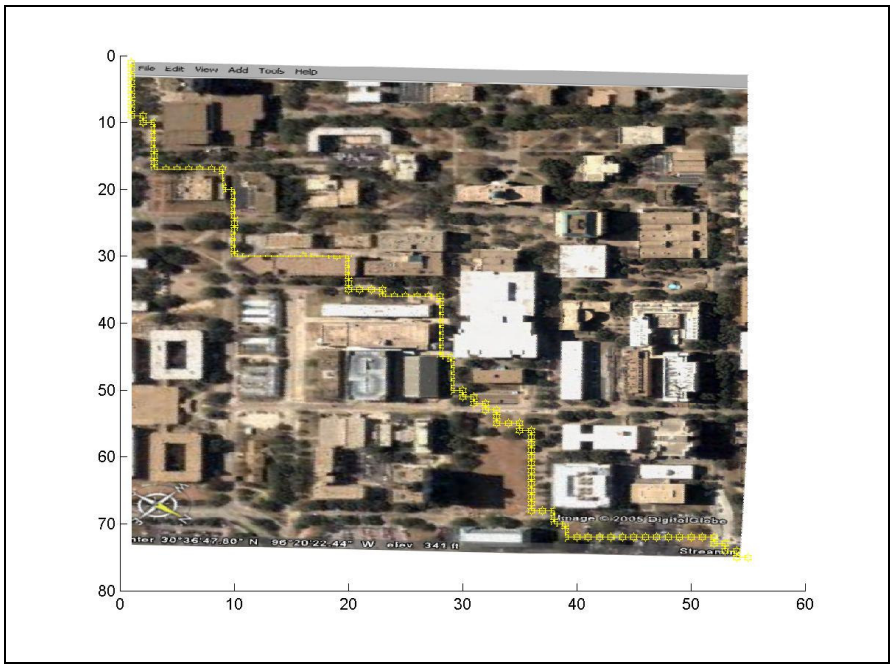

Fig. 3 Top View of Navigation through Campus

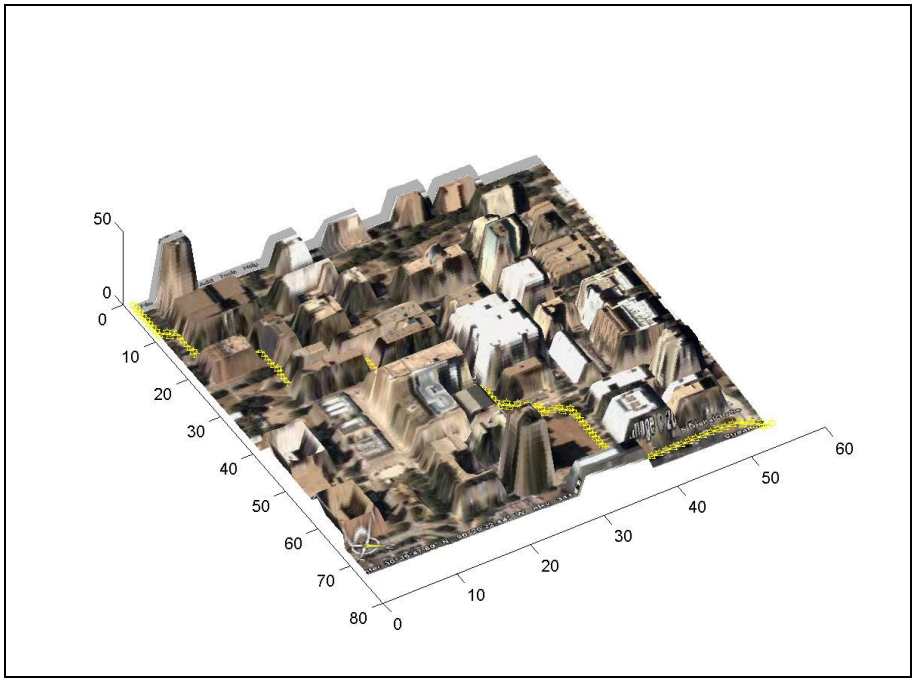

Fig. 4 Angled View of Navigation through Campus

D.2 Avoiding obstacles

The simulations in figures 3 and 4 used a static environment model. However, it is realistic to consider the situation where the environment model can be dynamic. If a newly sensed obstacle such as a vehicle or fallen building is encountered, the high-level flight controller should be able to adapt. When the helicopter radar senses a new obstacle, it creates a large transition cost so that the helicopter avoids flying into the newly sensed obstacle. In figure 5, the trajectory of the helicopter is plotted in yellow. Additional obstacles were added to the map after the helicopter started moving, so that the helicopter encountered a dynamic obstacle field and a blocked passage by the Texas A&M Library.

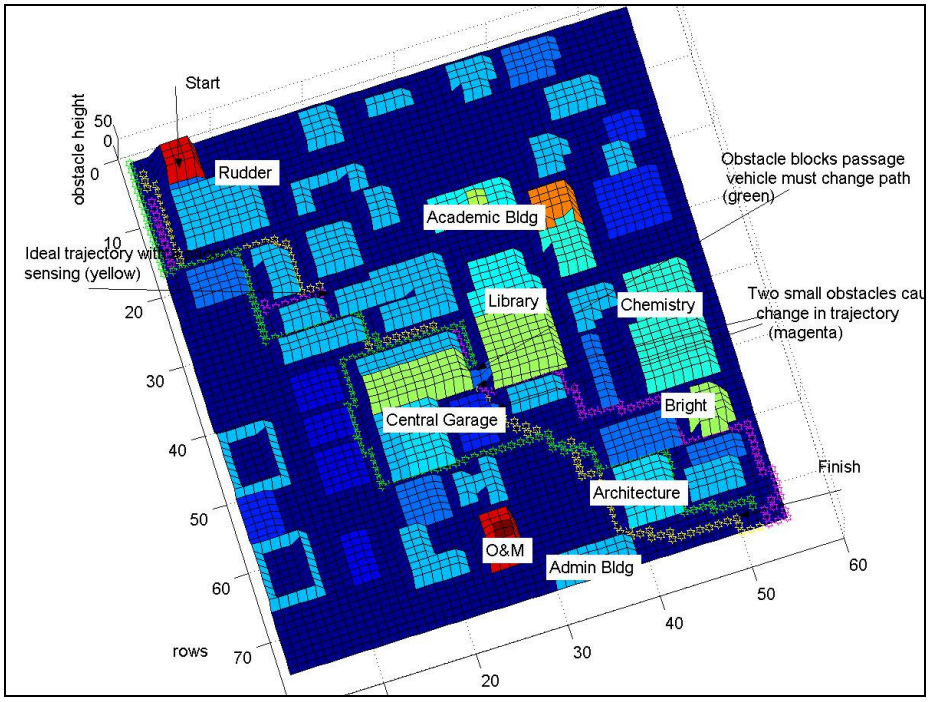

Fig. 5 Obstacles and Blocked Passage

D.3 Re-planning at a blocked passage

Occasionally, the vehicle may encounter an obstacle that entirely blocks a passage. The vehicle may not have time to recalculate the entire cost map J. In this thesis, updating the local region's cost map (a 10x10 grid) is sufficient for the vehicle to plan a new trajectory to get out of an impasse. It is conceivable that if the vehicle is unable to find a path out of a local minimum due to a blocked passage, then a global cost map may need to be recalculated. Figures 6 and 7 are included below to display the motion planner's response to blocked passages at various different locations on campus.

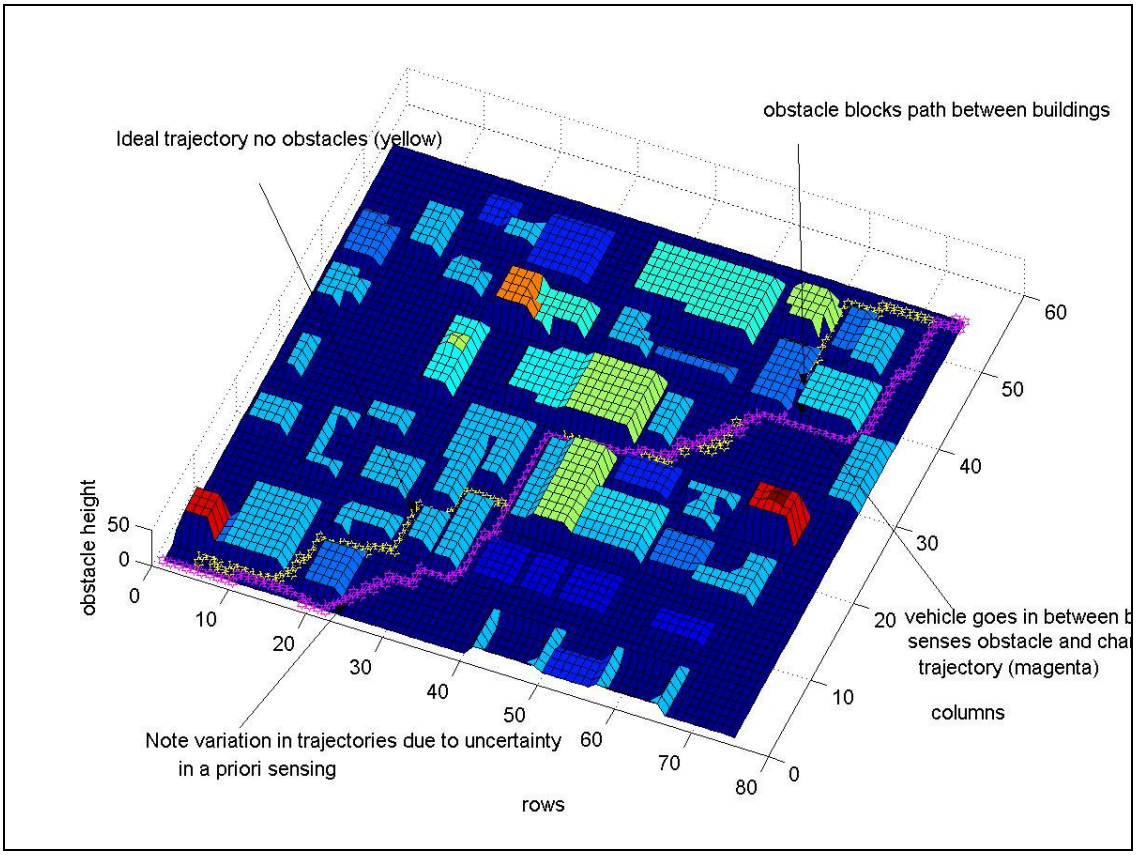

Fig. 6 Blocked Passage by Langford

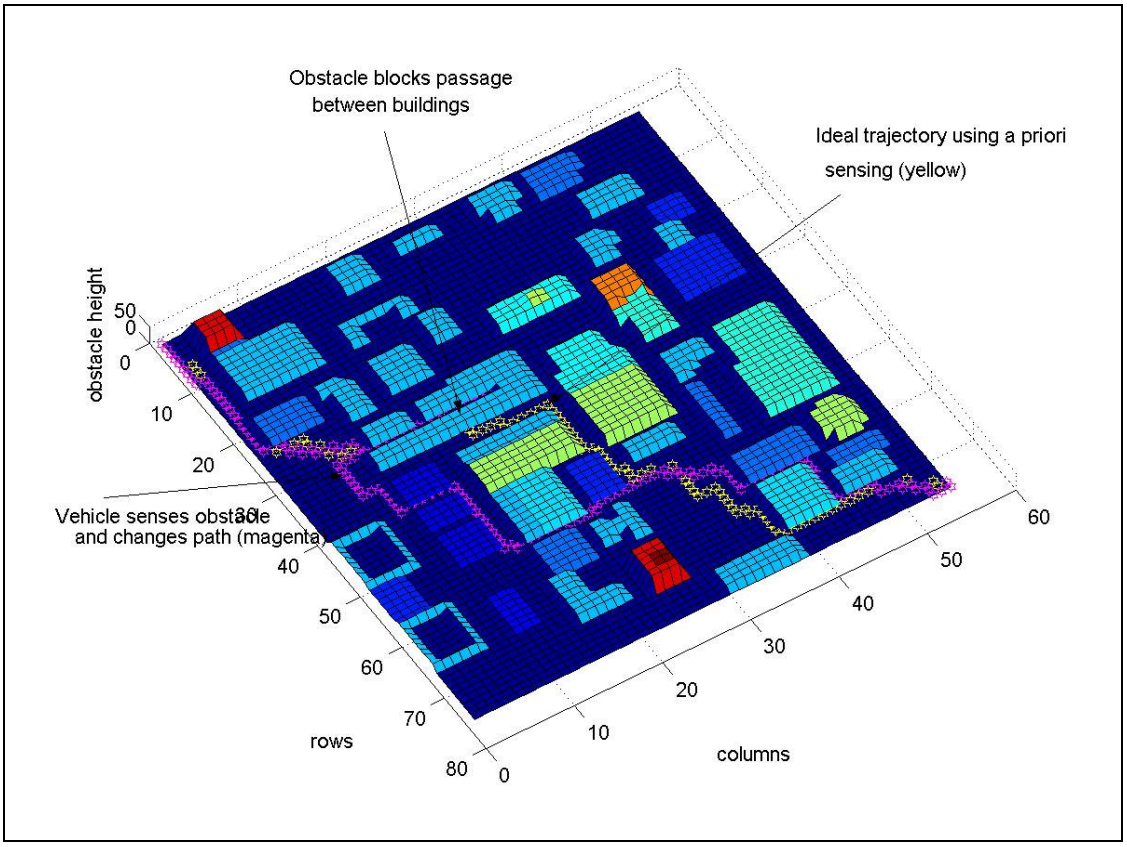

Fig. 7 Blocked Passage by Heldenfelds

D.4 Motion planning in the flight simulator

Multiple tests were run while the high-level motion planner and low-level flight controller were connected together. The high-level controller computation time was relatively minor compared to the time necessary to execute the waypoint commands. Running each MATLAB simulation took approximately 15-20 seconds, while a simulation of actually flying the helicopter to 180 waypoints, which were 10 meters apart, took approximately 12-15 minutes. If 180 waypoints were executed during the initial cost-to-go calculation then approximately $180*N*M\approx$ 3,000,000 major

computations were required, which requires a computation time of approximately (5-10 sec). During the MATLAB simulations the vehicle made a minimum of 1 local and 8 non-local sensor measurements and up to 4 local and 32 non-local sensor measurements per waypoint. This resulted in approximately 1800 optimizations run on environment probabilities and 720 waypoint calculations if 180 waypoints were executed. This requires a computation time of approximately 5-10 seconds. The following 8 plots (figures 8-15) demonstrate the performance of the motion planner and the flight simulator while connected together. Figures 8 and 9 were generated during the simulator testing 2 using the obstacle map of campus with a blocked passage by the library and a blocked passage by the pavilion. The obstacles were introduced after the helicopter started its flight. The vehicle does not always choose the same path. In this testing the vehicle did not take the anticipated route of going beside the library because it accounted for sensor uncertainties and planned a route by Dunn.

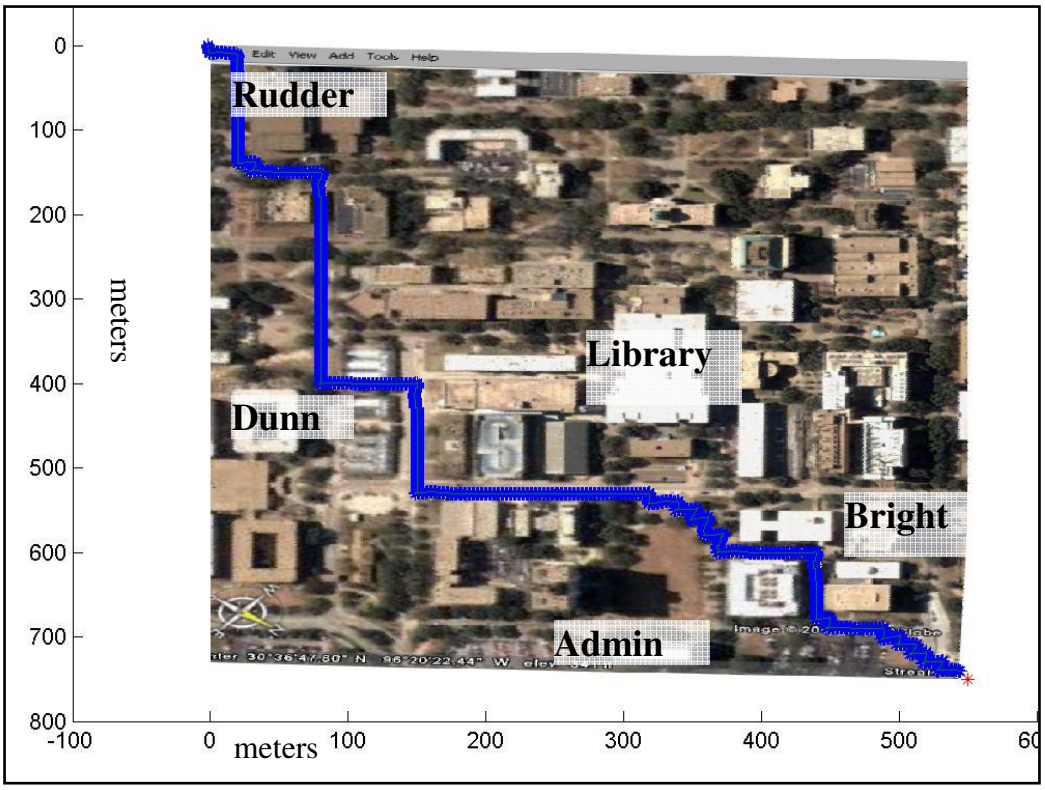

Fig. 8 Simulator Testing 2- Large View

Note in Figure 9 how closely the helicopter actual trajectory (blue line) follows the desired trajectory (yellow line).

Fig. 9 Simulator Testing 2- Detail View

Figures 10 and 11 were generated during the simulator testing 3 using the obstacle map of campus with a blocked passage by the library. The blocked passage by the library was introduced after the helicopter commenced its flight. As a result the helicopter sensed the blocked passage by the library and moved back and forth several times until it recalculated a local cost map. It took a few local cost map recalculations before the helicopter was able to determine a path out between the library and the library annex.

In a few simulations it was observed that a local cost map recalculation may not be sufficient to update the cost map and guide the helicopter out of a blocked passage. By only performing a local cost map update, it is possible to have a local minimum in which the helicopter is unable to navigate its way out of an area containing a newly sensed obstacle. The local cost map recalculations may need to be enlarged until the helicopter is able to find its way out of an area. If a local cost map is unable to update the cost map sufficiently for the helicopter to find its way out of an area, then it maybe necessary to perform an entire recalculation of the cost map. Since dynamic programming by definition is a global optimization, if the entire cost map is recalculated the helicopter will be able to determine the optimal path to the final destination.

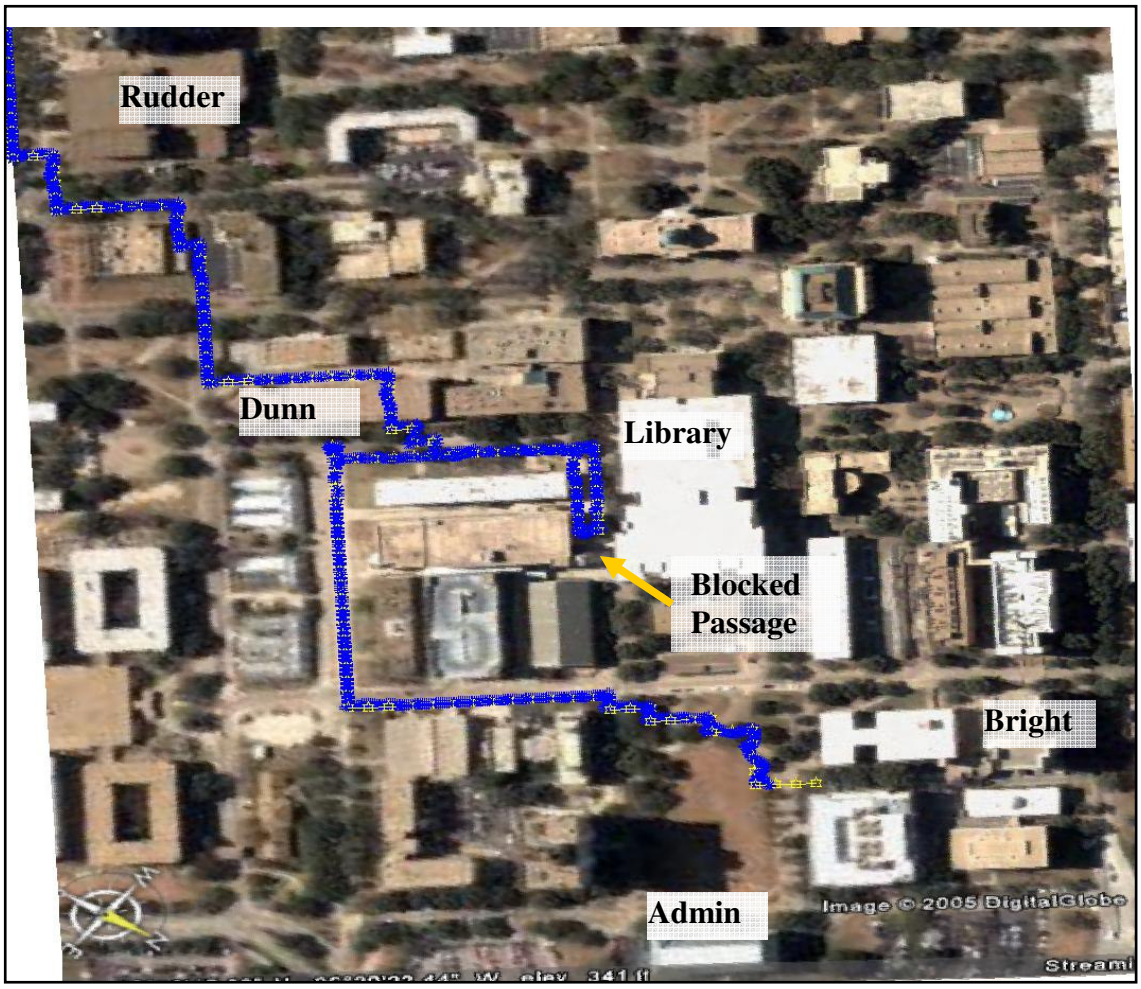

Fig. 10 Simulator Testing 3- Large View

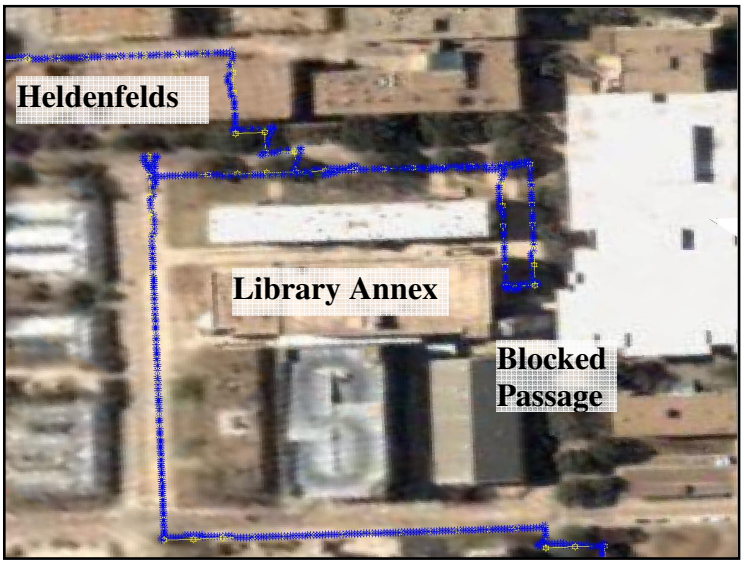

Fig. 11 Simulator Testing 3- Detail View

Figures 12 and 13 were generated during the simulator testing 4 using the obstacle map of campus with a blocked passage by Heldenfelds and Langford. The blocked passages and two obstacles near the library were introduced. The helicopter only encountered one of the obstacles and the result is represented on figure 13.

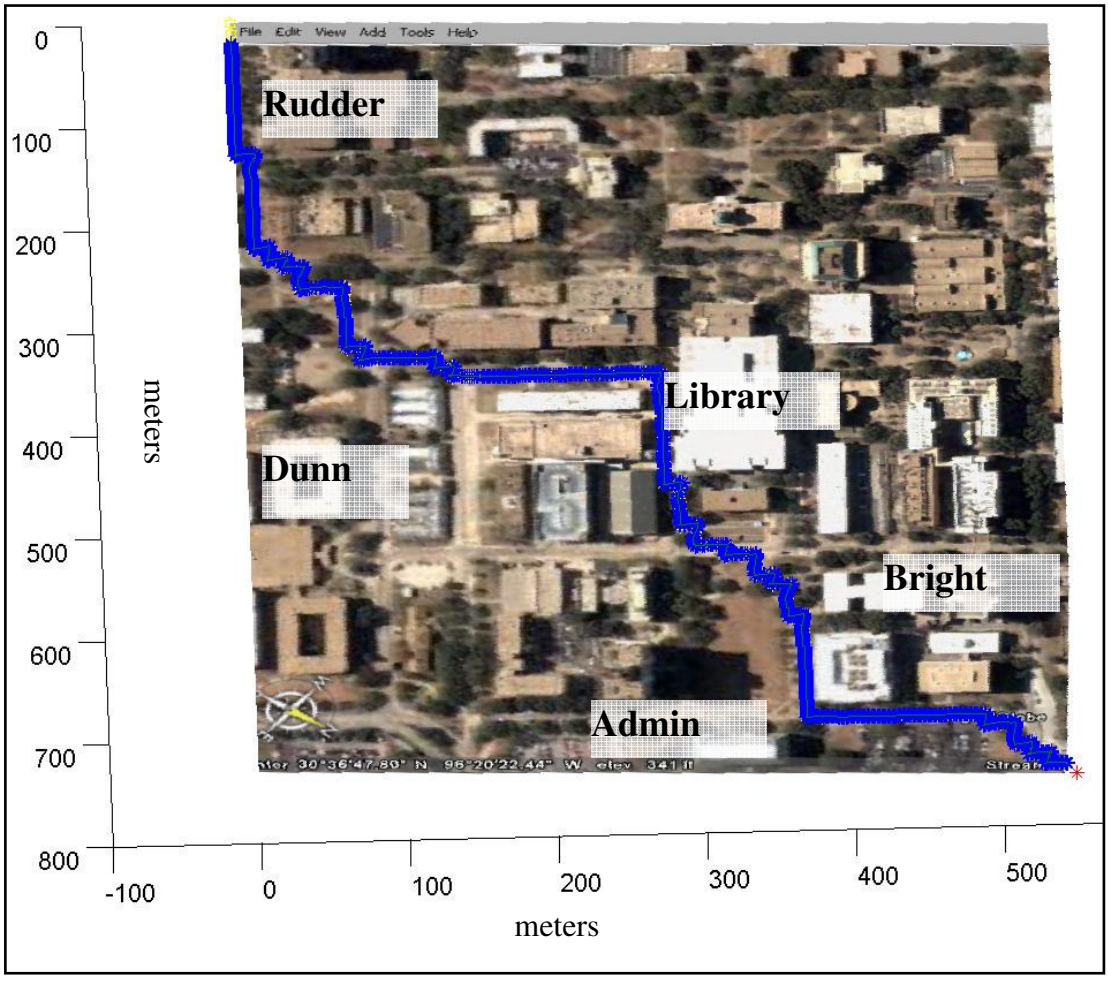

Fig. 12 Simulator Testing 4- Large View

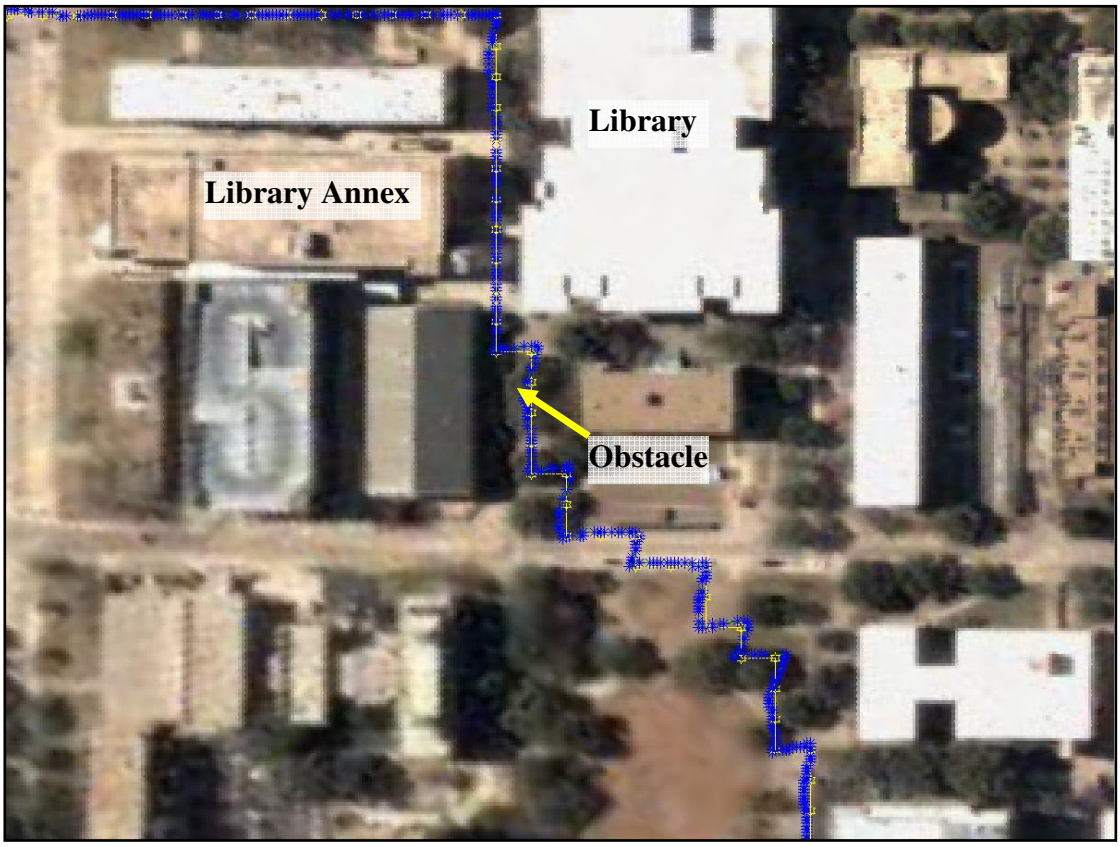

Fig. 13 Simulator Testing 4- Detail View

Figures 14 and 15 were generated during the simulator testing 5 using the obstacle map of campus with a blocked passage by Heldenfelds and Langford. The helicopter only encountered the blocked passage by Heldenfelds. After several recalculations of the local cost map the helicopter was able to leave the boxed in area.

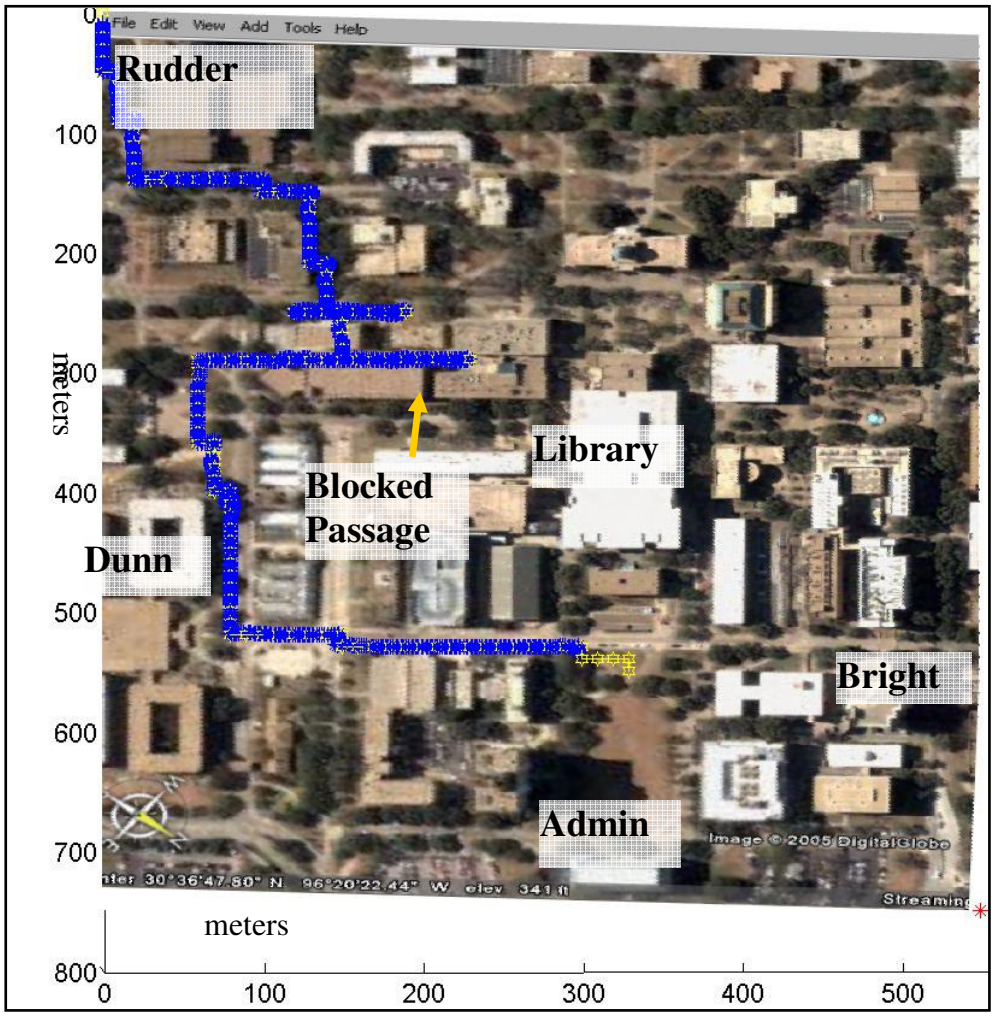

Fig. 14 Simulator Testing 5- Large View

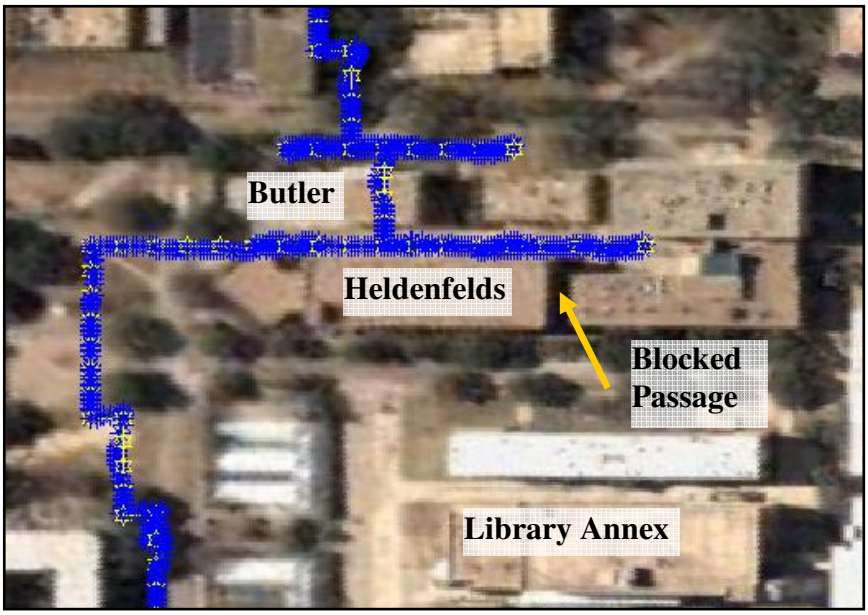

Fig. 15 Simulator Testing 5- Detail View

5. CONCLUSION

In this work, a methodology was presented for intelligent exploration of a partially known environment using a non-local sensor. It was shown that a motion planning problem, under certain assumptions, can be reduced to the adaptive control of an uncertain Markov Decision Process, consisting of a known control dependent system state and an unknown control independent environment. The feasibility of the planning methodology was illustrated by testing on an unmanned helicopter navigating through an urban environment in a six DOF flight simulation.

The frequency of the cost-to-go update is of considerable interest. At one end of the spectrum, the entire cost-to-go map could be updated at every time instant, which might be computationally infeasible, while at the other end of the spectrum, only an initial cost-to-go map could be calculated before the navigation begins. However, both these extremes are possibly not "optimal" and the best solution might be somewhere midway. It was surmised that the cost-to-go may need to be changed when the environments, which are sensed after the helicopter begins its flight, start looking "significantly different" from the estimates developed according to a priori data. However, these are qualitative statements and need to be quantified in terms of algorithms.

Another area of interest is to consider implementing heuristics in the dynamic programming algorithm that would allow for an accelerated computation of the cost-to-go map by initially excluding the evaluation of states that are far from the desired path of the helicopter. Also, the controller architecture used in this research only uses the high-

level motion planner to detect and avoid obstacles, while the low-level controller is assumed to track the desired trajectories closely enough to ensure obstacle avoidance. A low-level controller needs to be designed that guarantees the local avoidance of obstacles while tracking the high-level motion plans, so that integration of the high-level motion planner and low-level flight controller can result in a truly intelligent autonomous system.

Further, the amalgamation of the methodology presented in this thesis with existing motion planning methods such as Probabilistic Roadmaps or D* might lead to more robust, near real-time implementable motion planning algorithms.

REFERENCES

[1] Chakravorty, S., and Junkins, J. L., "Intelligent Exploration of Unknown Environments with Vision Like Sensors," *Proceedings of the 2005 IEEE/ASME International Conference of Advanced Intelligent Mechatronics*, Monterey, CA, 2005, pp. 1204-1209.

[2] Chakravorty, S., and Junkins, J. L., "A Methodology for Intelligent Path Planning," *Proceedings of the 2005 IEEE International Symposium on Intelligent Control*, Limassol, Cyprus, 2005, pp. 592-597.

[3] Kumar, P. R., and Varaiya, P. *Stochastic Systems: Estimation, Identification and Adaptive Control*, Prentice-Hall, Englewood Cliffs, NJ, 1986.

[4] Borkar, V., and Varaiya, P., "Adaptive Control of Markov Chains, I: Finite Parameter Set," *IEEE Transaction on Automatic Control*, Vol. 24, No. 6, 1979, pp. 953-958.

[5] Mandl, P., "Estimation and Control in Markov Chains," *Advances in Applied Probability*, Vol. 6, 1974, pp. 40-60.

[6] Bertsekas, D. P., and Tsitsiklis, J. N., *Neuro-Dynamic Programming,* Athena Scientific, Belmont, MA, 1996.

[7] Sutton, R. S., and Barto, A. G., *Reinforcement Learning: An Introduction,* MIT Press, Cambridge, MA, 1998.

[8] Sutton, R. S., Barto, A. G., and Williams, R. J., "Reinforcement Learning Is Direct Adaptive Optimal Control," *IEEE Control Systems Magazine*, Vol. 12, No. 2, 1992, pp. 19-22.

[9] Bellman, R. E., *Dynamic Programming*, Princeton University Press, Princeton, NJ, 1957.

[10] Bellman, R. E., and Dreyfus, S. E. *Applied Dynamic Programming,* Princeton University Press, Princeton, NJ, 1962.

[11] Bertsekas, D. P., *Dynamic Programming and Optimal Control*, Athena Scientific, Belmont, MA, 2000.

[12] D. E. Kirk *Optimal Control Theory: An Introduction*, Prentice-Hall, Inc., Englewood Cliffs, NJ, 1970.

[13] Lozano-Pérez, T. "Spatial Planning: A Configuration Space Approach," *IEEE Transactions on Computers,* Vol. C-32, No. 2, 1983, pp. 108-120.

[14] Latombe, J. C., *Robot Motion Planning,* Kluwer, Boston, MA, 1991.

[15] Latombe, J. C., Lazanas, A., and Shekhar, S., "Robot Motion Planning with Uncertainty in Control and Sensing," *Artificial Intelligence*, Vol. 52, 1991, pp.1-47.

[16] Mason, M. T., "Automatic Planning of Fine Motions: Correctness and Completeness," *Proceedings of IEEE Conference on Robotics and Automation*, Scottsdale, AZ, 1989, pp. 484-489.

[17] Lavalle, S. M., "Robot Motion Planning: A Game-Theoretic Foundation," *Algorithmica*, Vol. 26, 2000, pp. 430-465.

[18] Thrun, S. "A Probabilistic On-Line Mapping Algorithm for Teams of Mobile Robots," *The International Journal of Robotic Research*, Vol. 20, No. 5, May 2001, pp. 335-363.

[19] Thrun, S. "Probabilistic Algorithms in Robotics," *AI Magazine*, Vol. 21, No. 4, 2000, pp. 93-109.

[20] Burgard, W., Fox, D., Jans, H., Matenar, C., and Thrun, S., "Sonar-Based Mapping of Large-Scale Mobile Robot Environments Using EM," *Proceedings of the International Conference on Machine Learning*, Bled, Slovenia, 1999, pp. 67-76.

[21] Castellanos, J. A., Montiel, J. M., Neira, J., and Tardos, J. D., "The SP Map: A Probabilistic Framework for Simultaneous Localization and Mapping," *IEEE Transactions on Robotics and Automation*, Vol. 15, 1999, pp. 948-953.

[22] Dissanayake, G., Durant-White, H., and Bailey, T., "A Computationally Efficient Solution to the Simultaneous Localization and Mapping Problem," *ICRA'2000 Workshop W4: Mobile Robot Navigation and Mapping*, April 2000.

[23] Hu, H., and Brady, M., "Dynamic Global Path Planning with Uncertainty for Mobile Robots in Manufacturing," *IEEE Transactions on Robotics and Automation*, Vol. 13, No. 5, October 1997, pp. 760-767.

[24] LaValle, S. M. *Planning Algorithms*, Cambridge University Press, New York, NY, 2006.

[25] Korf, R. E. "Artificial Intelligence Search Algorithms," *Algorithms and Theory of Computation Handbook*, CRC Press, Boca Raton, FL, 1999.

[26] Nilsson, N. J., *Principles of Artificial Intelligence*, Tioga Publishing Company, Wellsboro, PA, 1980.

[27] Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C., *Introduction to Algorithms* (2nd Edition), MIT Press, Cambridge, MA, 2001.

[28] Goodrich, M. T., Tammasia, R. *Algorithm Design: Foundations, Analysis and Internet Examples.* John Wiley & Sons, Inc., New York, NY, 2002.

[29] Dijkstra, E. W., "A Note on Two Problems in Connection with Graphs," *Numerische Mathematik,* Vol. 1, 1959, pp. 269-271.

[30] Pearl, J., *Heuristics: Intelligent Search Strategies for Computer Problem Solving*. Addison-Wesley, Boston, MA. 1985.

[31] Stentz, A. "The Focused D* Algorithm for Real-time Replanning," *Proceedings of the International Joint Conference on Artificial Intelligence,* Montreal, Canada, 1995, pp. 1652-1659.

[32] Stentz, A., and Herbert, M., "A Complete Navigation System for Goal Acquisition in Unknown Environments," *Autonomous Robots*, Vol. 2, No. 2, 1995, pp. 127-145.

[33] Hebert, M., McLachlan, R., and Chang, P. "Experiments with Driving Modes for Urban Robots," *Proceedings of the SPIE Mobile Robots*, Boston, MA, 1999, pp 140-149.

[34] Koenig, S., and Likhachev M., "Fast Replanning for Navigation in Unknown Terrain," *IEEE Transactions on Robotics,* Vol. 21, No. 3, 2005.

[35]Ghallab, M., Nau, D., and Traverso, P., *Automated Planning: Theory and Practice*, Morgan Kaufman, San Francisco, CA, 2004.

[36] Russell, S. and P. Norvig. *Artificial Intelligence: A Modern Approach,* (2nd Edition) Prentice-Hall, Englewood Cliffs, NJ, 2003.

[37] Kavraki, L. E., Svestka, P., Latombe, J. C., and Overmars, M. H., "Probabilistic Roadmaps for Path Planning in High-Dimensional Configuration Spaces," *IEEE Transactions on Robotics & Automation,* Vol. 12, No. 4, 1996, pp. 566-580.

[38] Song, G., Shawna, T., and Amato, N. "A General Framework for PRM Motion Planning," *Proc. IEEE International Conference on Robotics and Automation (ICRA),* , Taipei, Taiwan, 2003, pp. 4445-4450.

[39] Datta, A., Ho, M. T., and Bhattacharyya, S. P. *Structure and Synthesis of PID Controllers,* Springer-Verlag, New York, NY, 2000.

[40] Franklin, G. F., Powell, J. D., and Emami-Naeini, A., *Feedback Control of Dynamic Systems,* (4th Edition) Prentice-Hall, Inc., Upper Saddle River, NJ, 2002.

[41] Khalil, H. K., *Nonlinear Systems,* (2nd Edition) Prentice-Hall, Inc., Upper Saddle River, NJ, 1996.

APPENDIX A

UAV HELICOPTER

The helicopter used for this research was a remote control (RC) Bergen Observer with a Zenoah G-23 (23cc) engine. The helicopter weight after the addition of the flight controller was approximately 23 pounds and could still carry approximately 3 pounds of additional payload. The helicopter flight controller components are estimated to have a weight of approximately 7 pounds, this includes: aluminum skids, two battery packs, the Automated Flight Control System, D-Link (wireless network bridge), multiple wires, 2 RC receivers, and a relay board.

Fig. A.1 Josh Davis with the UAV Helicopter and Ground Station

A. UAV hardware

The RC Bergen Observer was provided by the Air Force Research Laboratory (AFRL) to Texas A&M in order to implement a Rotomotion Automated Flight Control System (AFCS). The helicopter was purchased and partially assembled by AFRL. The assembly before arrival of the equipment at Texas A&M included assembling the graphite composite (G10) frame, mounting the engine, installing camera mount, installing the tail boom, and assembling the rotor head. The helicopter was approximately fifty-percent assembled when it was delivered to Texas A&M in February 2005. The remaining assembly was performed at Texas A&M. This included purchasing additional parts such as ball links, servo connecting rods, fuel system, rotor blades, flybar paddles, and electronics equipment. The radio used was a Futaba 9CHP 9-Channel pulse code modulated (PCM) radio with four S9001 Servos (pitch, roll, throttle, collective) and Futaba GY401 Gyro with a S9254 digital servo for the tail rotor.

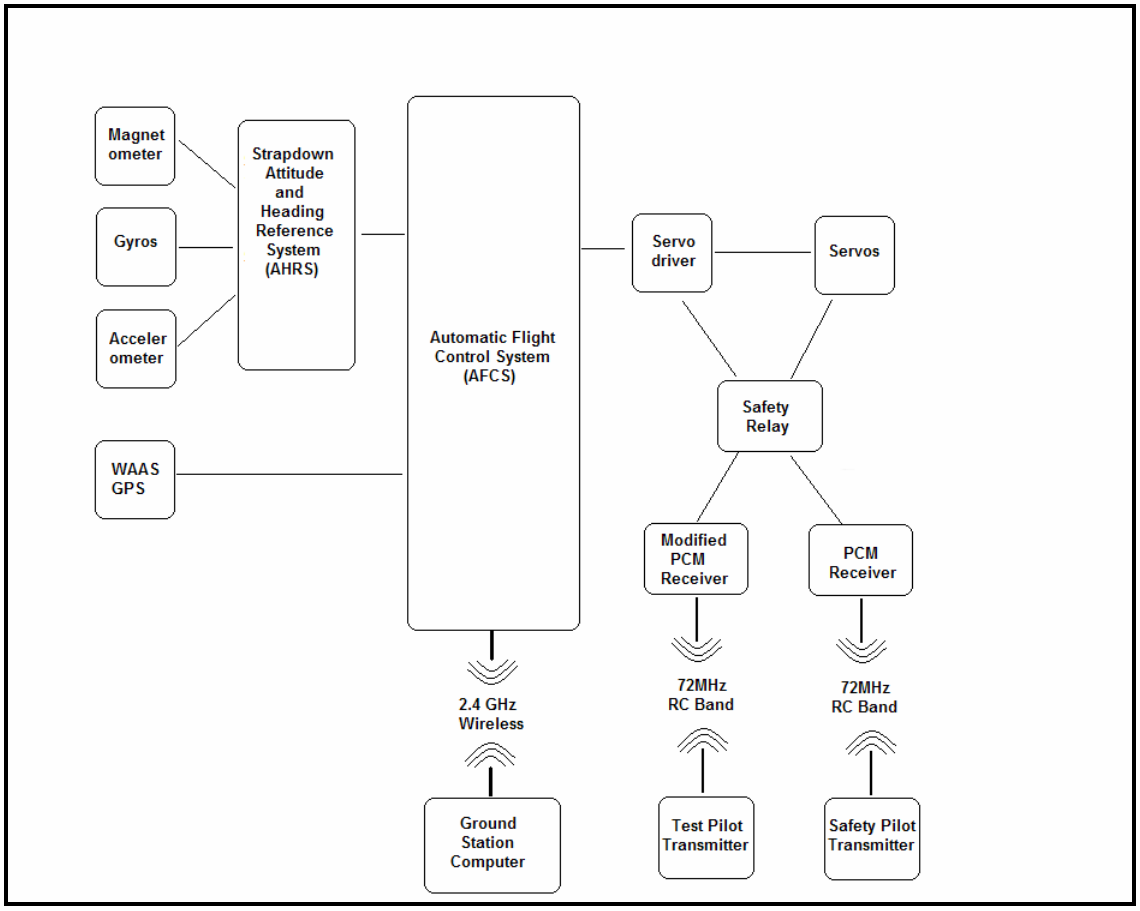

Fig. A.2 Controller Architecture

The AFCS has a six degree of freedom (6-DOF) inertial measurement unit (IMU) with three accelerometers and three gyros. The accelerometers were Analog Devices ADXL202AE. The accelerometers use a polysilicon structure to form a differential capacitor and measure ± 2 g static (gravitational) and dynamic accelerations. The gyros were Tokin CG-16D ceramic gyros with a piezoelectric ceramic column to detect a maximum angular rate of ± 90 degrees. The primary purpose of the accelerometers and gyros is to provide the helicopter with attitude information. The AFCS flight controller

also contains SENS-L magnetometers, which use an Application Specific Integrated Circuit (ASIC) architecture, from PNI Corporation's Magneto-Inductive (MI) product line. The three magnetometers are oriented orthogonal to each other to give the helicopter the current heading. Three magnetometers are used because the helicopter is not always level and ferrous helicopter components produce hard iron error.

Fig. A.3 Automated Flight Control System

A u-blox TIM-LP Global Positioning System (GPS) receiver sensor is used and implements differential updates using the Wide Area Augmentation System (WAAS). After using the WAAS differential updates, the position measurements are accurate to within 3 meters ninety-five percent of the time. The GPS sensor provides position measurements as well as velocity measurements by integrating the change in position. The AFCS includes a Mega 32 AVR Controller to decode the Futaba 1024 pulse code modulation (PCM) signal as well as perform analog to digital data conversion for the inertial measurement unit (IMU) sensors connected to the microcontroller. Information

is sent from the Mega 32 to an Intel XScale 400 processor. The Intel XScale 400Mhz processor runs the onboard flight computer and analyzes the state of the helicopter at a rate of 200 Hz.

B. UAV software

The Rotomotion software was developed in the Linux operating environment and includes a 3-dimensional flight simulator as well as a ground station. The simulations were performed using the Suse Linux 9.2 operating system on an Acer laptop with a 3.0 GHz Pentium 4 processor, 512 MB of Ram, and a 40GB hard drive. The Linux C/C++ software was developed for communication between the high-level flight controller in MATLAB and the Rotomotion low-level flight controller API. The C/C++ code was written and debugged using the Integrated Development Environment KDevelop 3.0.

The low-level flight controller, the AFCS, uses two Extended Kalman Filters for state estimation. One Kalman Filter is a 7 state filter: 4 states for Quaternion and 3 states for gyro bias. The other Kalman filter is a 6 state filter for North East Down (NED) Position and UVW Velocity. The position measurements are filtered to give a more accurate position hold. Also, the Doppler velocity from the GPS is integrated to smooth out abrupt changes in position.

The low-level flight controller also uses a series of nine Proportional Integral Derivative (PID) control loops to control the helicopter. The first three loops that must be tuned are the attitude control loops-roll (ϕ), pitch (θ), and yaw (ψ). These ensure that the helicopter achieves the proper attitude during its transition from a hover (stationary flight) to forward/sideway (translational flight) and yawing (rotational flight).

The second set of PID loops are the position control loops. These control loops ensure that the helicopter maintains a position–fore (x), side (y), and down (z). The final three PID loops, the velocity control loops, were developed to ensure steady velocities-fore ($\dot{x}$), side ($\dot{y}$), and down ($\dot{z}$).

The high-level flight controller uses dynamic programming to perform motion planning in MATLAB while running on a Linux operating system. The high-level flight controller performed a text file based inter-process communication between MATLAB and a program called "waypoint", which was developed using C/C++ in Kdevelop, to communicate with the flight controller's application programming interface (API).

The flight simulator operates on the same protocol as the AFCS. As a result the control algorithm can be tested in the flight simulator by executing the high-level motion planner to interface with the flight simulator rather than the AFCS. The interface between the waypoint program and the flight simulator (heli-sim) is essentially the same as the interface between the waypoint program and the AFCS. The primary difference is that flight simulator and the AFCS use different IP addresses.

The AFCS as well as the Rotomotion ground station operate using a Linux based operating system that was developed in the C/C++ programming language. The onboard flight controller runs a command line (non-graphical) version of Linux. The graphical interface of the ground station developed Rotomotion, LLC. uses the Fast Light ToolKit (FLTK) software, which is an open source C/C++ software based upon the OpenGL standard for computer graphics. Fast Light ToolKit is used to implement an event-

driven graphical interface so the ground station can monitor the helicopter's status as well as issue commands.

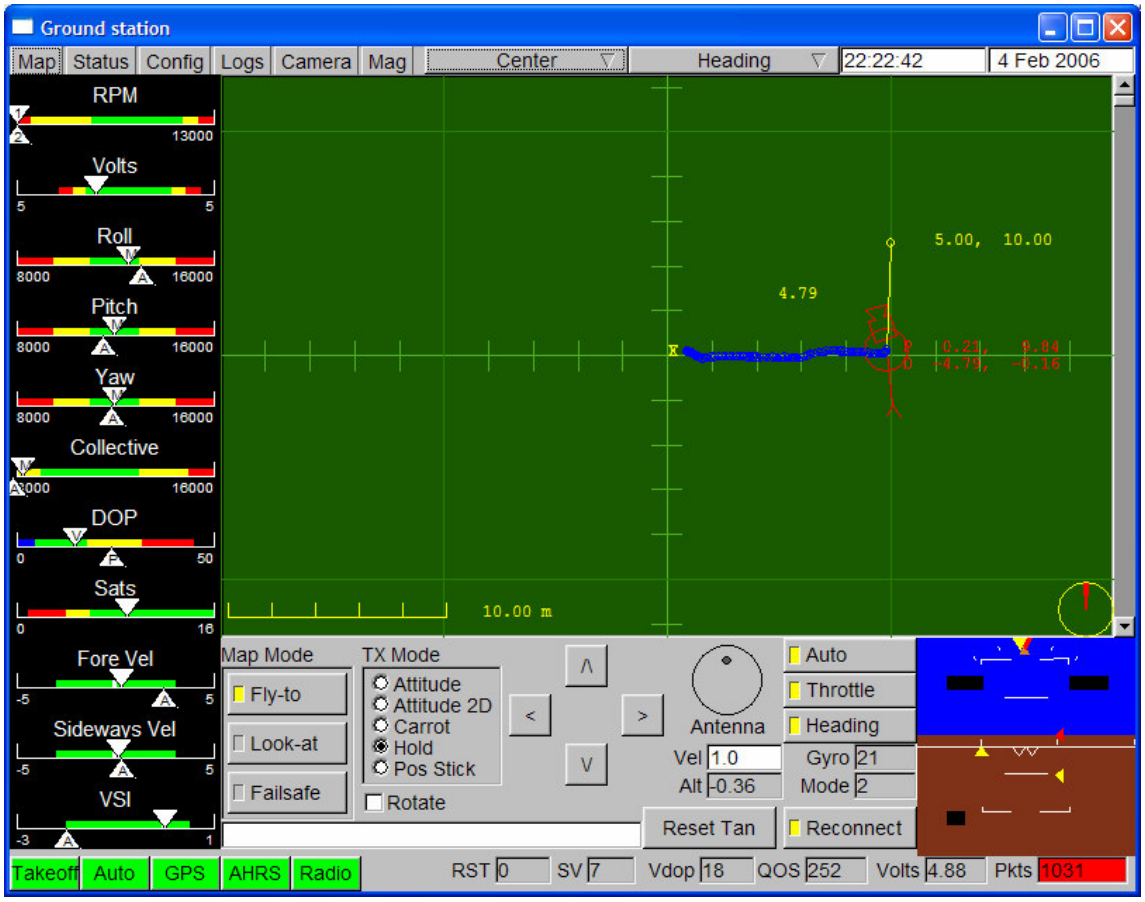

Fig. A.4 Ground Station Interface During Hardware in Loop Testing

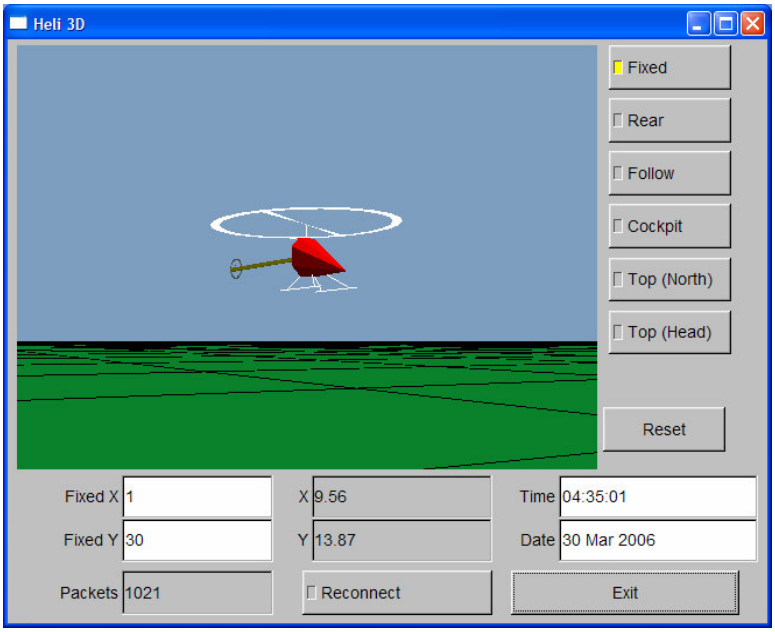

Fig. A.5 Heli-3d Viewer for Ground Station

C. Flight testing and hardware in the loop testing

In May 2005, flight testing included several maintenance flights to ensure proper remote control performance and prepare for integration of the Rotomotion AFCS. During testing, the engine was tuned with the proper low and high-end throttle settings as well as the throttle-collective performance curve. The tracking of the rotor blades was adjusted to ensure proper flight performance and radio equipment was tested.

The helicopter wiring was changed so that the RC flight commands could be executed through the AFCS in June 2005. A safety pilot transmitter was installed which used a relay to bypassing the flight controller if necessary and was capable of sending servo commands directly to the servos from a separate transmitter. In July 2005, the helicopter was taken to Charleston, South Carolina to tune the flight controller. The flight controller was previously used on a helicopter airframe of relatively similar size so the

gains in the controller reasonably held the helicopter in its position. It was still necessary to fine tune the gains for the new airframe.

Several remote control and autonomous flights were performed with the unmanned helicopter. The flight testing challenges included electronic difficulties from electro-magnetic interference and loss of communication, as well as mechanical difficulties from clutch slippage and bearing malfunctions. After the flight testing, several hardware-in-the-loop tests were performed to demonstrate that the high-level motion planning algorithm could be implemented on the flight hardware.

During the hardware-in-the-loop testing, motion planning algorithms were executed outside the Flight Mechanics Laboratory hanger on Texas A&M's Riverside Campus. The helicopter was pushed on a cart to simulate the helicopter flying to various waypoints. All of the electronics and telemetry were running and only the helicopter engine remained off. A motion planning algorithm was executed and the helicopter was moved on a cart to a series of waypoints as telemetry was logged by the ground station and the control surfaces were monitored for proper deflection. After successful hardware-in-the-loop testing, the remainder of the motion planning algorithm testing and development was performed in the 6 DOF Rotomotion, LLC. flight simulator.

VITA

Joshua Daniel Davis graduated from Auburn University with a Bachelor of Science in Mechanical Engineering in 2003. He worked for the Air Force Research Laboratory and later accepted a position as a graduate assistant research in the Department of Aerospace Engineering at Texas A&M. He completed his Master of Science in Aerospace Engineering in 2006 and accepted a position as an engineer with Bell Helicopter in Hurst, Texas. Joshua Daniel Davis can be reached through his advisor:

Joshua Daniel Davis

c/o Dr. Suman Chakravorty

3141 TAMU- Aerospace Engineering

College Station, Texas 77843-3141