Motion Planning Under Uncertainty: Application to an Unmanned Helicopter

Joshua D. Davis[1] and Suman Chakravorty[2]
Department of Aerospace Engineering, Texas A&M University, College Station, TX 77843

A methodology is presented in this work for intelligent motion planning in an uncertain environment using a non-local sensor, such as a radar. This methodology is applied to an unmanned helicopter navigating a cluttered urban environment. It is shown that the problem of motion planning in an 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 significantly reduce the computational burden of the associated algorithms. The methodology is applied to the problem of motion planning for an unmanned helicopter through a partially known model of the Texas A&M campus and the testing is done on a flight simulator.

I. Introduction

IN this paper, 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 helicopter navigates the obstacles in the environment. The motion planner involves a high-level planner which plans against the uncertainty in the environment and issues its commands as 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. However, the methodology is applicable for any non-local sensor. 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*

[1] Graduate Assistant Research, Department of Aerospace Engineering, 3141 TAMU.
[2] Assistant Professor, Department of Aerospace Engineering, 3141 TAMU, Member, AIAA.

corresponds to the *(x,y,z)* coordinates of the helicopter 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 paper, 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 without 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. 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.[11] 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 still "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.[12] 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.[13] In the past decade, there has been an increasing interest in the case when the environment in which the system is operating is partially or completely unknown. The uncertainty in the environment is treated as a deterministic worst case[14,15] or on a probabilistic average case basis.[16] In "probabilistic robotics", there has been substantial research in the localization of a mobile robot while simultaneously mapping the environment.[17,18,19,20,21] In recent years, 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.[22] The area of artificial intelligence in the past tended to focus on problem solving in a discrete state space.[23,24] 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 breadth 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.[25] Depth first search considers the cost from a starting point to the goal state along a feasible path and may later consider alternate paths.[26] 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.[27]

One of the most well known graph search methods for motion planning is A* 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 in a static environment but requires complete recalculation if the environment changes.[28] 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*),[29] 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,[30] urban robots, and even the Mars rover.[31] 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.[32]

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.[33,34] 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.[35] The roadmap is a graph of randomly generated collision-free paths, which are connected by some simple fast planning method.[36] 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. It is noted here that the method of implementing the higher level planner are different for different methods, for instance, see reference[37]. However, the underlying philosophy of these methods is the same: the higher level planner plans at the global environmental level based on some coarse uncertain knowledge of the environment, while the local planner implements the global plans making sure that it avoids the obstacles and constraints arising due to the incomplete knowledge of the environment. 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 paper is organized as follows. Section 2 details the structure of the motion planner. Section 3 contains the formulation of the high level planner 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.

II. 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 the sequence of waypoints 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 the environment and determines the next waypoint. This process continues until the goal state is achieved. Figure 1 below outlines the basic motion planning architecture.

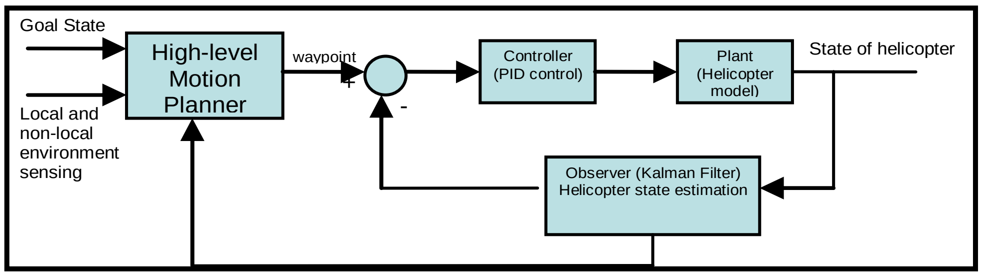
Goal State
High-level Motion Planner
waypoint
+
-
Controller (PID control)
Plant (Helicopter model)
State of helicopter
Local and non-local environment sensing
Observer (Kalman Filter) Helicopter state estimation

Fig. 1. Control 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.[38,39] 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. 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 accounts for the uncertainties at shorter time/length scaling, and also for the uncertainties in the system dynamical model. In the following, we shall use the terms planner and controller interchangeably, the two terms are synonymous in the context of this paper.

III. Planning Under Uncertainty Using Non-Local Sensing

First, we recount some results that will be required for the motion planning.[1,2] The development in the rest of this section is exclusively for the higher level planner.

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, 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). 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).

We assume that the state set S is discrete, i.e., the planning domain has been gridded into a finite number of cells. From hereon assume that the system state is sensed perfectly and only the environment is sensed imperfectly (in the context of the previous section, it means that the particular grid that the vehicle is in, is known with certainty). 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. The control of the high level planner is discrete and corresponds to the adjacent grid cell that the UAV needs to move to, given its current location cell. 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.,*

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

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.*

The second assumption above essentially means that the environmental random process (note that the environment is modeled as a random process here) is temporally stationary and spatially uncorrelated/ independent. Deterministic environments (like an unstructured terrain) automatically satisfy the above assumptions.

Proposition 1 *Under assumptions A 3.1, 3.2, the following holds:*

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

The above result is a direct consequence of assumptions 3.1 and 3.2 , please see Chakravorty and Junkins [1,2] for more details. 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 (please see Bertsekas [11] for more details) :

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

(3)

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 β < 1 is a given discount factor.

The following environment sensing model is adopted:

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.

Associated with every observation-vantage point pair, *(q(s),s′), s′∈ 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′* ∈ *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.

B. Estimation

Consider the following relationship:

$$\pi(\hat{q}(s)) = \frac{1}{\pi\left(F\left(s\right)\right)} \sum_{s' \in F(s), q(s)} p\left(\hat{q}(s)/q\left(s\right), s'\right) p\left(q\left(s\right)\right) \pi\left(s'\right) \tag{4}$$

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 (or periodicity) 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)*.

Note that the above equation can be derived heuristically by considering $\pi(\hat{q}(s))$ as a probability and then conditioning it on random variables $q\left(s\right)$ and 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 4. Mathematically:

$$\pi_t(\hat{q}_n(s)) := \frac{1}{t}\sum_{n=1}^{t} 1(\hat{q}_n(s) = \hat{q}(s)) \tag{5}$$

$$\pi_t(s) := \frac{1}{t}\sum_{n=1}^{t} 1(s_n = s) \tag{6}$$

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:

$$P_t(s) := \arg\min_{P\in\bar{V}} \left\|\Pi_t(s) - \Gamma_t(s)P\right\|^2 \tag{7}$$

where

$$P_t(s) = \left[p_t(q_1(s)),\dots,p_t(q_D(s))\right]' \tag{8}$$

$$\Pi_t(s) = \left[\pi_t(\hat{q}_1(s)),\dots\pi_t(\hat{q}_D(s))\right]' \tag{9}$$

$$\Gamma_t(s) = \lambda(s)\left[\gamma_t^{ij}(s)\right] \tag{10}$$

$$\gamma_t^{ij}(s) = \sum_{s'\in F(s)} p(q_i(s)/q_j(s),s')\pi_t(s') \tag{11}$$

||.|| denotes the Euclidean norm in $\Re^D$, and $\bar{V}$ represents the space of all probability vectors in $\Re^D$. Note that we call the algorithm recursive because the quantities $\Pi_t(s)$ and $\Gamma_t(s)$ are obtained in a recursive fashion. 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 $\Gamma_t(s)$ stochastic, i.e., such that the sum of the elements of every column of the matrix is unity.

C. Control

Consider the stochastic optimal control problem posed in equation 3. 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:

$$u^*(s,q(s)) = \arg\min_u \sum_{(r,\overline{q}(r))} p(r/s,u)p(\overline{q}(r))[c((r,\overline{q}(r),(s,q(s)),u) + \beta J^*(r,\overline{q}(r))], \quad (12)$$

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:

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

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 7. 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

$$\mu_k\left(s,q(s)\right) = \arg\min_u \sum_{(r,\overline{q}(r))} p\left(r/s,u\right) p_{t_k}\left(\overline{q}(r)\right)\left[c\left(\left(r,\overline{q}(r)\right),\left(s,q(s)\right),u\right) + \beta J_k(r,\overline{q}(r))\right] \quad (14)$$

where

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

Next, a few of the salient properties of the control problem posed are listed. The results are given without proof, please see references [1,2] for more details about the proofs. 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 A3.1-A3.2, 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:

$$J(s,q(s)) = \min_u \sum_{(r,\overline{q}(r))} p(r/s,u)p(\overline{q}(r))[c((r,\overline{q}(r),(s,q(s)),u) + \beta J(r,\overline{q}(r))] \tag{15}$$

Let

$$\overline{c}\left(s,u,q\left(s\right)\right) = \sum_{\left(r,q'(r)\right)} p\left(r/s,u\right)p\left(\overline{q}\left(r\right)\right)c\left(\left(r,\overline{q}\left(r\right)\right),\left(s,q\left(s\right)\right),u\right). \tag{16}$$

Noting that

$$p((r,\overline{q}(r))/(s,q(s)),u) = p(r/s,u)p(\overline{q}(r)), \tag{17}$$

it follows that

$$J\left(s,q(s)\right) = \min_u \left[\overline{c}\left(s,u,q\left(s\right)\right) + \beta\sum_r p\left(r/s,u\right)\overline{J}(r)\right], \tag{18}$$

where

$$\overline{J}\left(s\right) = \sum_{q(s)} p\left(q\left(s\right)\right)J\left(s,q\left(s\right)\right). \tag{19}$$

Noting that

$$u^*\left(s,q\left(s\right)\right) = \arg\min_u \left[\overline{c}\left(s,u,q\left(s\right)\right) + \beta\sum_r p\left(r/s,u\right)\overline{J}\left(r\right)\right], \tag{20}$$

it can be concluded that an average value of the cost-to-go function $J(s,q(s))$, at the system state s, namely $\overline{J}(s) = \sum_{q(s)} p(q(s))J(s,q(s))$, 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 $\overline{J}(s)$. In order to answer this question, note that

$$\overline{J}\left(s\right) = \sum_{q(s)} p\left(q\left(s\right)\right)\min_u \left[\overline{c}\left(s,u,q\left(s\right)\right) + \beta\sum_r p\left(r/s,u\right)\overline{J}(r)\right]. \tag{21}$$

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

$$\overline{T}\ \overline{J}\left(s\right) = \sum_{q(s)} p\left(q\left(s\right)\right)\min_u \left[\overline{c}\left(s,u,q\left(s\right)\right) + \beta\sum_r p\left(r/s,u\right)\overline{J}(r)\right], \forall s. \tag{22}$$

The following proposition states that the "average DP operator", $\overline{T}$, 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.[37] This allows us to significantly reduce the computational burden of the planning algorithm. Hence, we have the following result.

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

IV. Testing on an Unmanned Helicopter

The methodology presented thus far in this paper was used for motion planning of an unmanned helicopter in an uncertain environment. The motion planning algorithms were implemented in six DOF simulations of an unmanned 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. At any given location the helicopter had four possible high-level control actions: Go (Forward/Back/Right/Left) to the adjacent grid point.

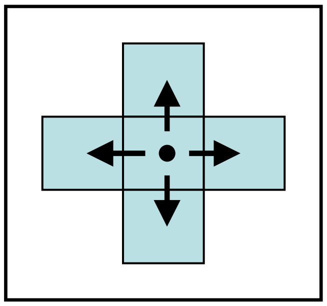

Fig. 2. Possible high-level control actions

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 67ms (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 a 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 primary 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 $\bar{J}(s)$, 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 the environment model (in terms of the environmental probabilities) at 8 adjacent grid-points. If the helicopter is commanded in a direction that has not been sensed by the radar sensor, the helicopter changes its direction to the commanded direction, and performs new measurements of the environment states. This is because the radar only has 90 degree field of vision, and thus, cannot sense all directions at once. 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. 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 but hits obstacles, while the value of $\alpha = 6$ draws the vehicle to the final destination without hitting 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:

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

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 $\alpha d\left(r\right)$ represents the tuning parameter α multiplied by the distance of the future state, *r,* to the goal state.

The results shown in figures 3 through 15 are for 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 50%, 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 50% to 90%, 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. Each sensor measurement was the true value corrupted by a Gaussian noise term based upon the distance to the obstacle. The simulations using the Eaton VORAD model had a 95% accuracy for the non-local state sensing.

The simulations were also performed with various different uncertainty models so that the effects of using sensors, with different accuracy levels, on the performance of the algorithm could be quantified. For example, in figures 5, 6, and 7 the sensor model has an accuracy of 96% for the local sensing, an accuracy of 93% at a distance of 10 meters, and an accuracy of 91% at a distance of 14 meters. The accuracy of the radar for a given range, was found by linearly interpolating between the accuracy of the sensor at 1 meter (99%) and 100 meters (57%). For a sensor uncertainty of 50%, the vehicle performed as if no sensor data was available. For sensor uncertainties between 70% and 80%, the performance increased significantly. For uncertainties in the range of 90% t0 99%, the 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 A&M campus.

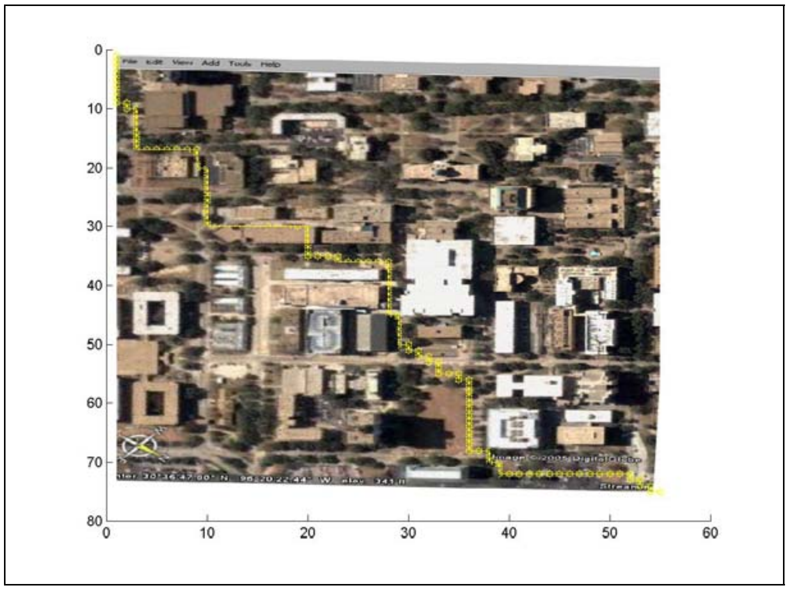

Fig. 3. Top view of navigation through campus

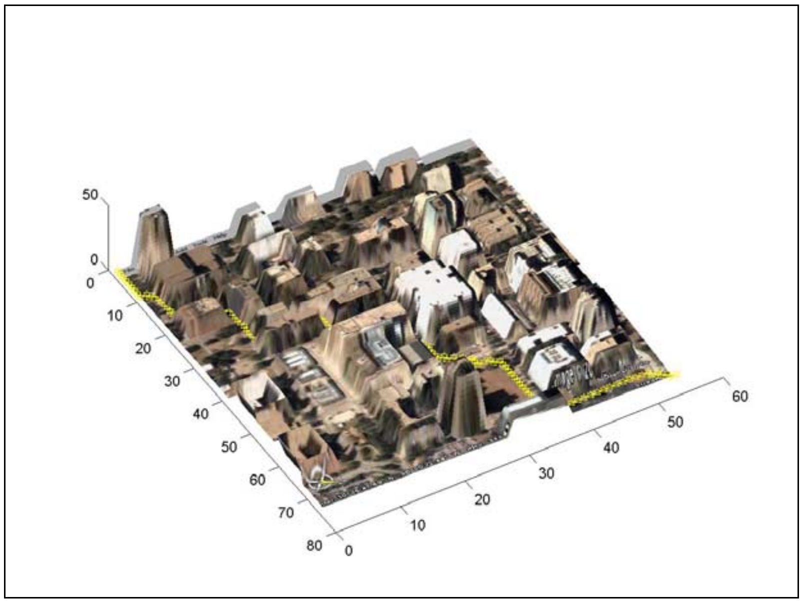

Fig. 4. Angled view of Navigation through campus

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 is 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 to the change. 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. For instance, in figures 5-7, additional obstacles were added to the map after the helicopter started moving, so that the helicopter encountered a dynamic obstacle field and blocked passages at various points on the A&M campus. At the algorithmic level, this alters the cost structure of the DP algorithms and thus, the helicopter has to replan when such an event occurs, i.e., re-evaluate the cost-to-go map of the entire planning domain.

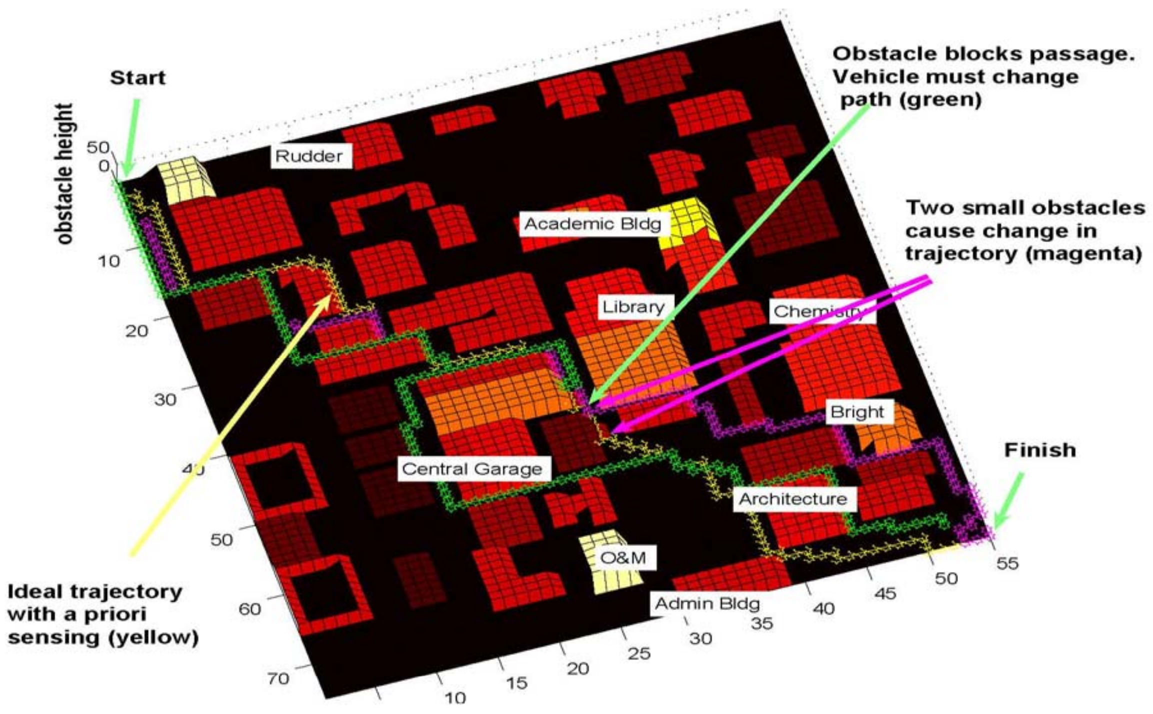
Start
obstacle height
50
0
10
20
30
40
50
60
70
10
15
20
25
30
35
40
45
50
55
Rudder
Academic Bldg
Library
Chemistry
Bright
Central Garage
Architecture
O&M
Admin Bldg
Obstacle blocks passage. Vehicle must change path (green)
Two small obstacles cause change in trajectory (magenta)
Finish
Ideal trajectory with a priori sensing (yellow)

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, i.e., the new optimal cost-to-go for the entire environment. Practically, updating the local region's cost map (a 10x10 grid) is sufficient for the vehicle to replan its trajectory in order to get out of such an impasse. This is a heuristic modification of the theoretical framework which makes it practically implementable. The size of the local region to update is a heuristic choice and is addressed experimentally. In this case, updating the local 10x10 grid was sufficient. It is conceivable that if the vehicle is unable to find a path out of a local impasse due to a blocked passage, then the global cost-to-go map may need to be recalculated. Figure 5-7 display the motion planner's performance with respect to blocked passages at various different locations on campus. In figure 5, the a priori designed trajectory is shown in yellow. Two different obstacles are then considered in the plot. In the first case, a major obstacle blocks the passage between the library and the central garage, which makes the helicopter reverse its direction once it senses the obstacle and find an alternate route (shown in green). In the second case, there are two small obstacles blocking the passage which results in the vehicle changing its route suitably (shown in magenta). In figure 6, there is an obstacle between the architecture and the Bright building which causes the vehicle to reverse its direction and find a way around the architecture building (shown in magenta). In figure 7, an obstacle blocks the vehicle path in between two buildings. As in the previous cases, the vehicle senses this obstacle and reverses its path to find a way around it.

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).

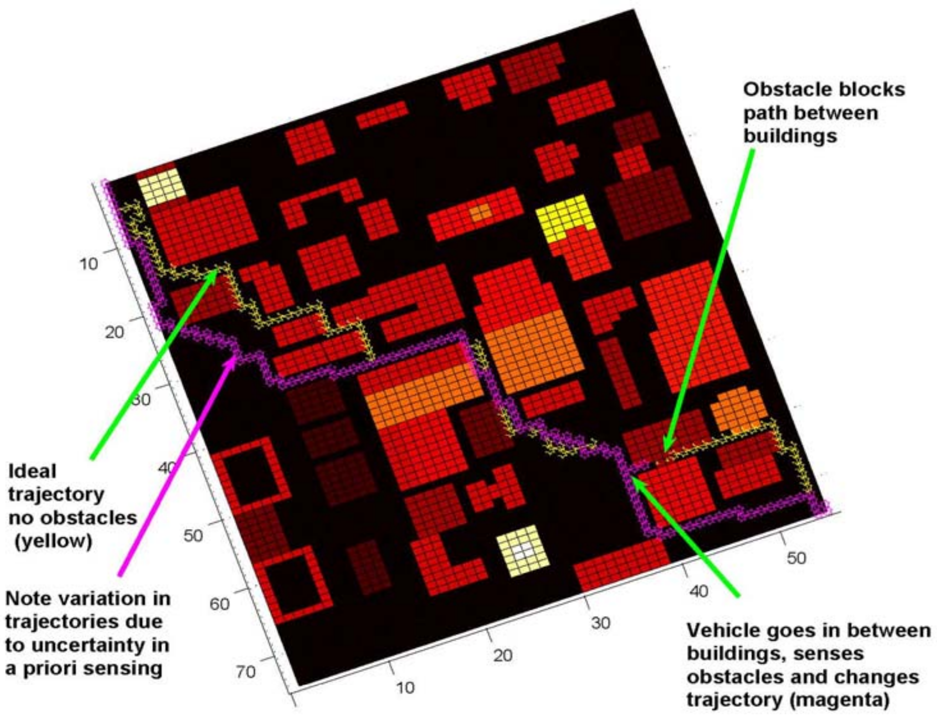
Obstacle blocks path between buildings
Ideal trajectory no obstacles (yellow)
Note variation in trajectories due to uncertainty in a priori sensing
Vehicle goes in between buildings, senses obstacles and changes trajectory (magenta)
10
20
30
40
50
60
70
10
20
30
40
50

Fig. 6. Blocked passage by Langford

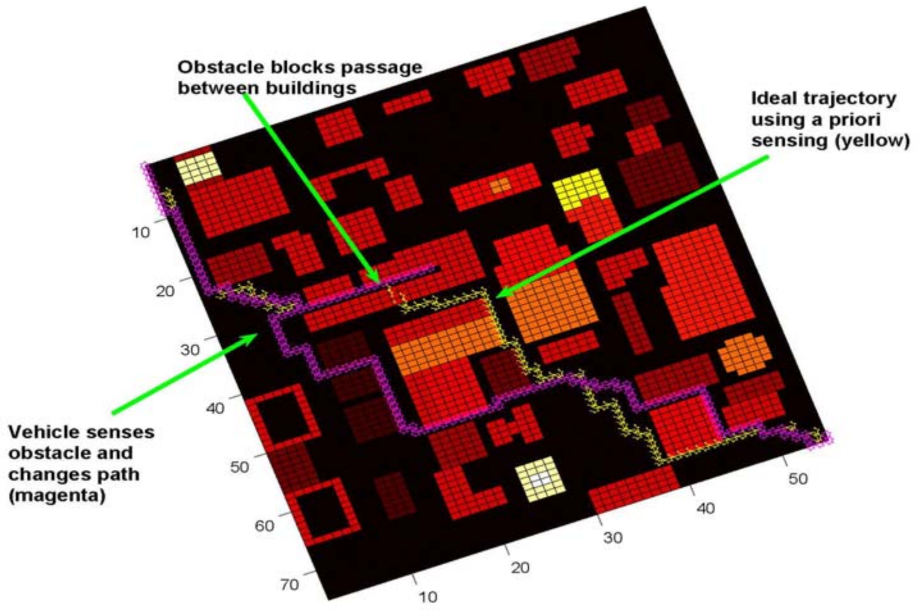
Obstacle blocks passage between buildings
Ideal trajectory using a priori sensing (yellow)
Vehicle senses obstacle and changes path (magenta)
10
20
30
40
50
60
70
10
20
30
40
50

Fig. 7. Blocked passage by Heldenfelds

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 optimization runs for the various environment probabilities and 720 waypoint calculations if 180 waypoints were executed. This requires a computation time of approximately 5-10 seconds. Figures 8-15 demonstrate the performance of the motion planner and the flight simulator while connected together. Figure 8 and 9 were generated during simulator testing 2 using the campus map along with a fictitious 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.

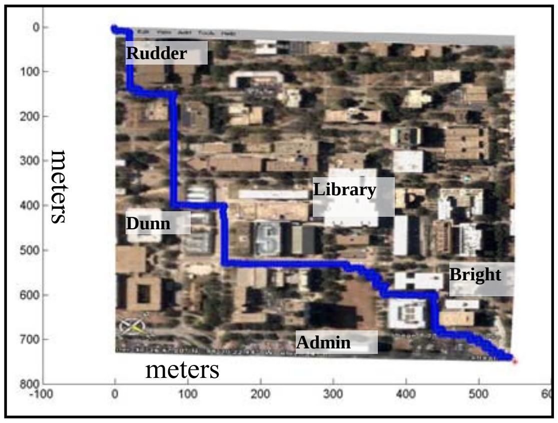

Fig. 8. Simulator testing 2-large view

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

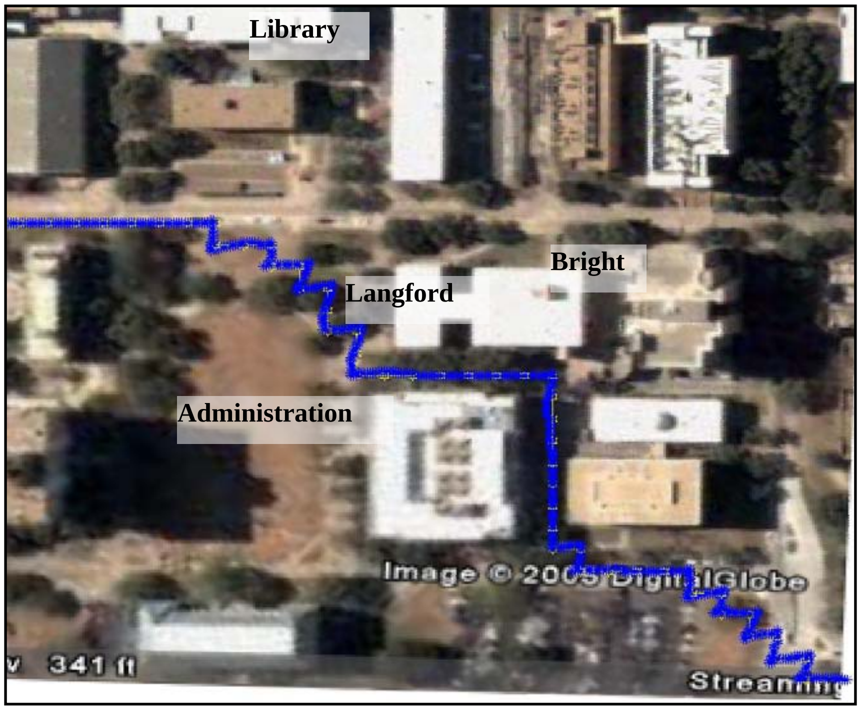

Fig. 9. Simulator testing 2- detail view

Figure 10 and 11 were generated during simulator testing 3 using the campus map with an a priori unknown blocked passage by the library. 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 several 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 get trapped in a local minimum. 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.

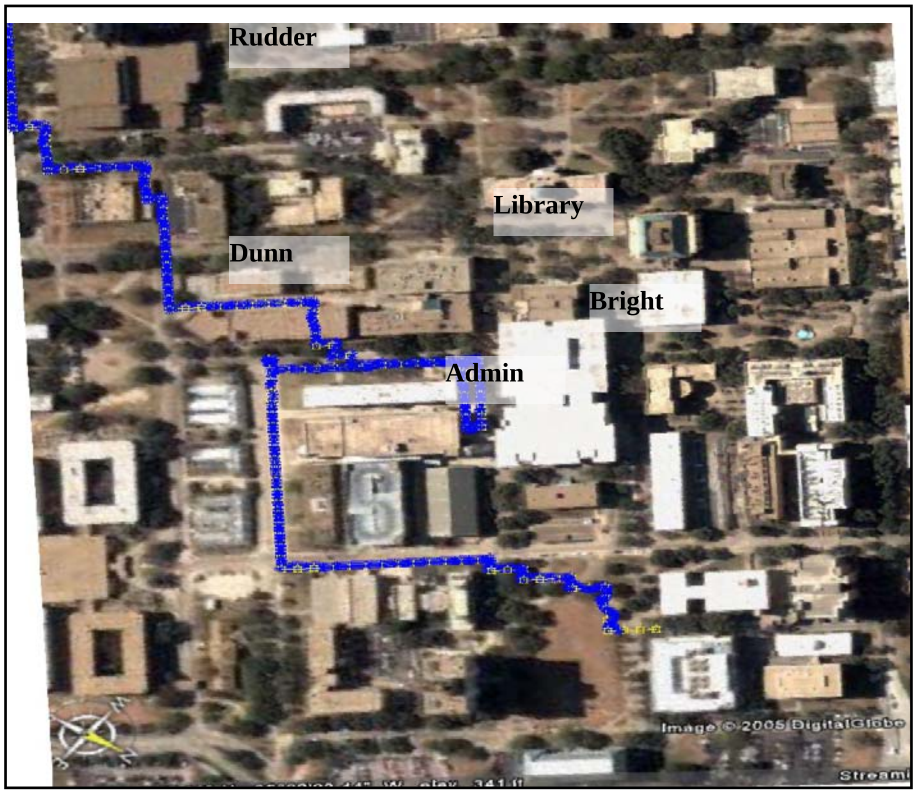

Fig. 10. Simulator testing 3- large view

Fig. 10 Includes a blocked passage by the library in which the helicopter recalculated the local cost map to exit.

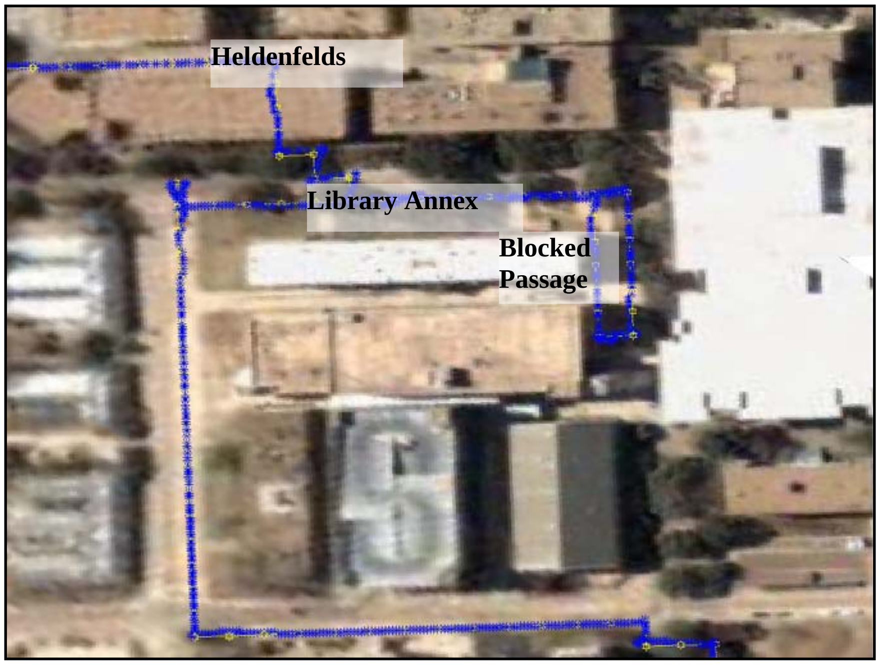

Fig. 11. Simulator testing 3- detail view

Figure 12 and 13 were generated during 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 after the helicopter started its flight. The helicopter only encountered one of the obstacles and the result is represented on figure 13.

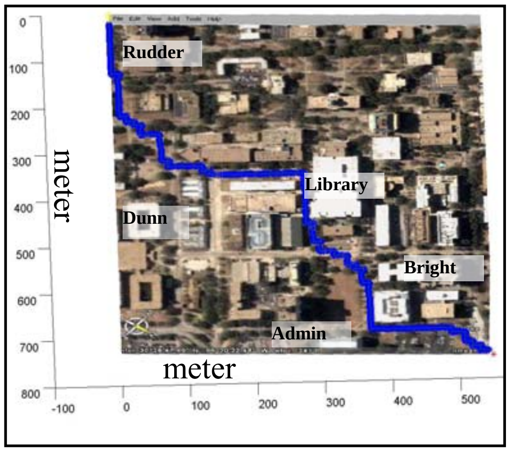

Fig. 12. Simulator testing 4-large view

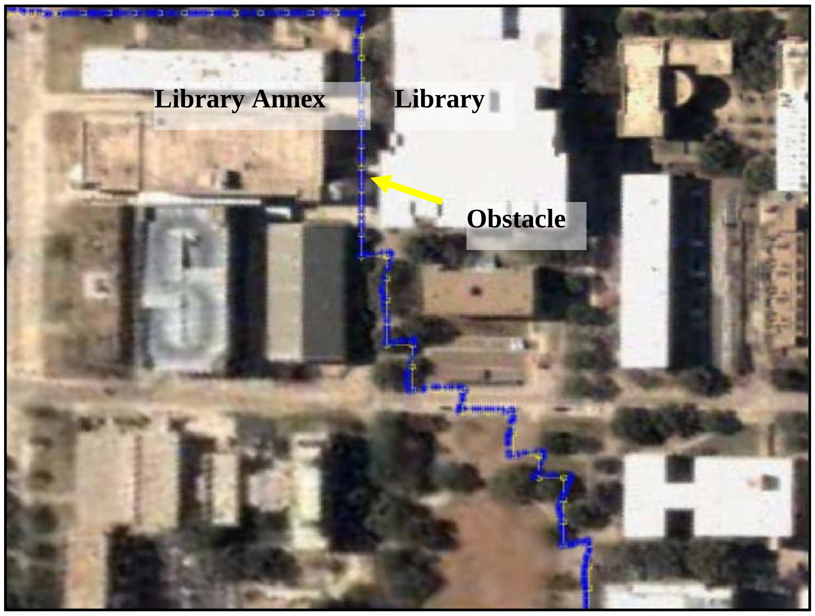

Fig. 13. Simulator testing 4- detail view

Figure 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.

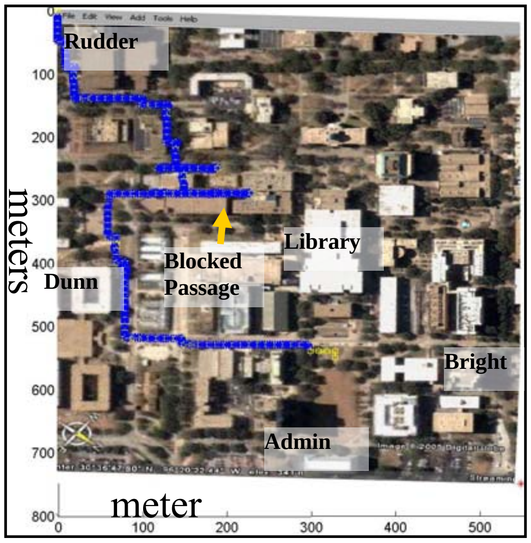

Fig. 14. Simulator testing 5-large view

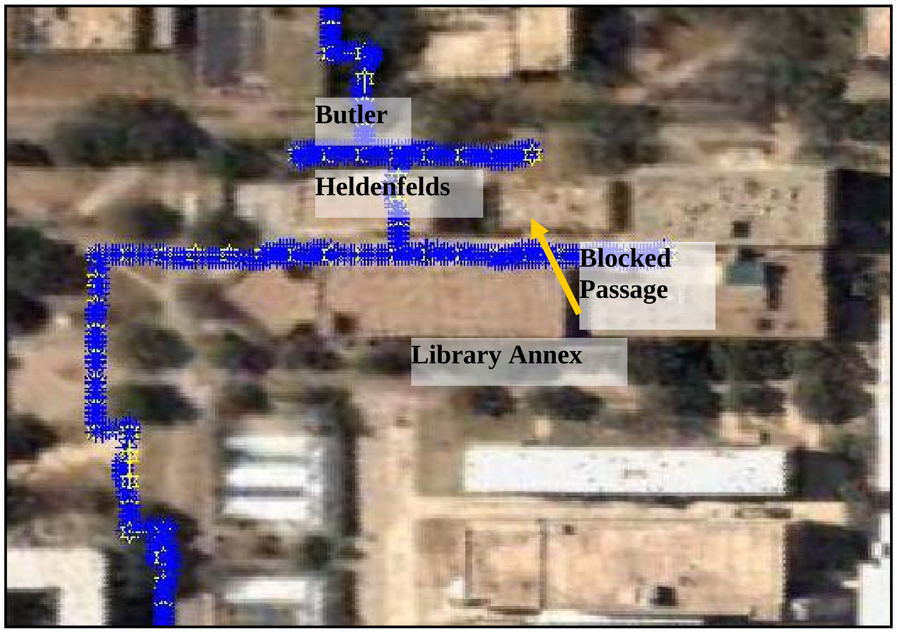

Fig. 15. Simulator testing 5-detail view

V. 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 paper 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*, IEEE Publications, Piscataway, NJ, 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*, IEEE Publications, Piscataway, NJ, 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 Transactions 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] D. E. Kirk, *Optimal Control Theory: An Introduction*, Prentice-Hall Inc., Englewood Cliffs, NJ, 1970.

[12] Lozano-Pérez, T. "Spatial planning: A configuration space approach," *IEEE Transactions on Computers,* Vol. C-32, No. 2, 1983, pp. 108-120.

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

[14] 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.

[15] Mason, M. T., “Automatic Planning of Fine Motions: Correctness and Completeness”, *Proceedings of IEEE Conference on Robotics and Automation*, IEEE Publications, Piscataway, NJ , 1989, pp. 484-489.

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

[17] 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.

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

[19] 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*, Morgan-Kaufmann, San Francisco, 1999, pp. 23-29.

[20] 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.

[21] 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*, IEEE Publications, Piscataway, NJ, April 2000.

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

[23] Korf, R. E., “Artificial Intelligence search algorithms,” *Algorithms and Theory of Computation Handbook*, CRC Press, Boca Raton, FL, 1999.

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

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

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

[27] Dijkstra, E. W., “A note on two problems in connection with graphs,” *Numerische Mathematik,* Vol. 1, 1959, pp. 269-271.

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

[29] Stentz, A. “The focused D* algorithm for real-time replanning,” *Proceedings of the International Joint Conference on Artificial Intelligence,* Morgan Kaufmann, San Francisco, 1995, pp. 1652-1659.

[30] 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.

[31] Hebert, M., McLachlan, R., and Chang, P., "Experiments with driving modes for urban robots," *Proceedings of the SPIE*, vol. 3838, 1999, pp. 70-78.

[32] Koenig, S., and Likhachev M., "Fast Replanning for Navigation in Unknown Terrain", *IEEE Transactions on Robotics,* Vol. 21, No. 3, 2005, pp. 354-363.

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

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

[35] 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.

[36] Song, G., Shawna, T., and Amato, N. "A General Framework for PRM Motion Planning" *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA),* IEEE Publications, Piscataway, NJ, 2003, pp. 4445-4450.

[39] Khalil, H. K., *Nonlinear Systems, 2nd Edition*, Prentice-Hall, Inc., Upper Saddle River, NJ, 1996, pg. 64.