THREE-DIMENSIONAL TRAJECTORY GENERATION FOR FLIGHT WITHIN AN OBSTACLE RICH ENVIRONMENT

By

RYAN DONOVAN HURLEY

A THESIS PRESENTED TO THE GRADUATE SCHOOL
OF THE UNIVERSITY OF FLORIDA IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE

UNIVERSITY OF FLORIDA

2009

Dedicated to Jessica

ACKNOWLEDGMENTS

I would first like to express my sincerest gratitude to Dr. Rick Lind for not only providing council for my research, but for giving me a chance to pursue a profession that would not have been possible without his assistance. His advise has always been for my own benefit, even if I did not want to believe it sometimes. I would also like to thank Dr. Joe Kehoe and Abe Pachikara for their assistance in this research. In addition, I would like to thank Dr. Prabir Barooah and Dr. Anil Rao for taking the time to sit on my committee and review my research.

Thanks are also in order to my fellow members of the Flight Control Lab. I could not ask for a better work environment or a better group of friends. Thanks go out to (in clockwise order) Michael, Baron, Sebastian, David, Stephen, Brian, Rob, Sanketh, Dan, and Dong for (in random order) impressing my wife, impressing me, making me laugh all the time, always being the butt of the joke, tweeting, also impressing my wife, letting me know how you feel, being the good distraction, moving me out of the way, and grounding us all.

Recognition is also in order to all of my friends and family for all the encouragement they have provided me. I want to especially thank my parents Fred and Joan Hurley for always loving me no matter what I do. They never want to take credit but they made me who I am today. Additionally, I want to thank my father and mother-in-law, Brian and Christine Farrar, for the endless support they have provided me. God could not have given me a better family. I love you all.

Finally, to my motivator and spiritual guide, I want to thank my wife Jessica for all of her physical and mental support. I could not have done any of this without her.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Abstract of Thesis Presented to the Graduate School
of the University of Florida in Partial Fulfillment of the
Requirements for the Degree of Master of Science

THREE-DIMENSIONAL TRAJECTORY GENERATION FOR FLIGHT WITHIN AN OBSTACLE RICH ENVIRONMENT

By

Ryan Donovan Hurley

May 2009

Chair: Richard C. Lind Jr.
Major: Aerospace Engineering

Autonomous flight through urban environments requires methods to generate trajectories that traverse the region and its associated obstacles. This thesis introduces the development of a 3-dimensional motion planning algorithm using a random dense tree (RDT) based on a set of motion primitives in cooperation with a 3-dimensional version of the Dubins car called the Dubins airplane. The motion primitives consist of 3-dimensional maneuvers formulated as combinations of turn segments and straight segments with an associated constant rate of climb. The resulting motion planner builds the trajectory generating RDT by pruning nodes that intersect 3-dimensional obstacles while connecting the remaining nodes with the motion primitives. Several examples of the motion planner are presented for cases with no obstacles, building-style obstacles arranged in an urban environment, and an urban environment that includes bridges. These examples demonstrate that feasible paths are computed as sub-optimal solutions to minimize the cost of flight time.

CHAPTER 1
INTRODUCTION

1.1 Motivation

The maturation of micro air vehicles (MAVs) has introduced a class of aircraft with sufficient agility to enable flight through urban environments. An example of one of these MAVs is shown in Figure 1-1. Techniques for motion planning are required that can compute trajectories for such flight that acknowledge the fully 3-dimensional (3-D) nature of the mission and associated obstacles throughout the region. Also, tight tolerances on the flight path require that the resulting path is entirely feasible and can be accurately followed by the vehicle.

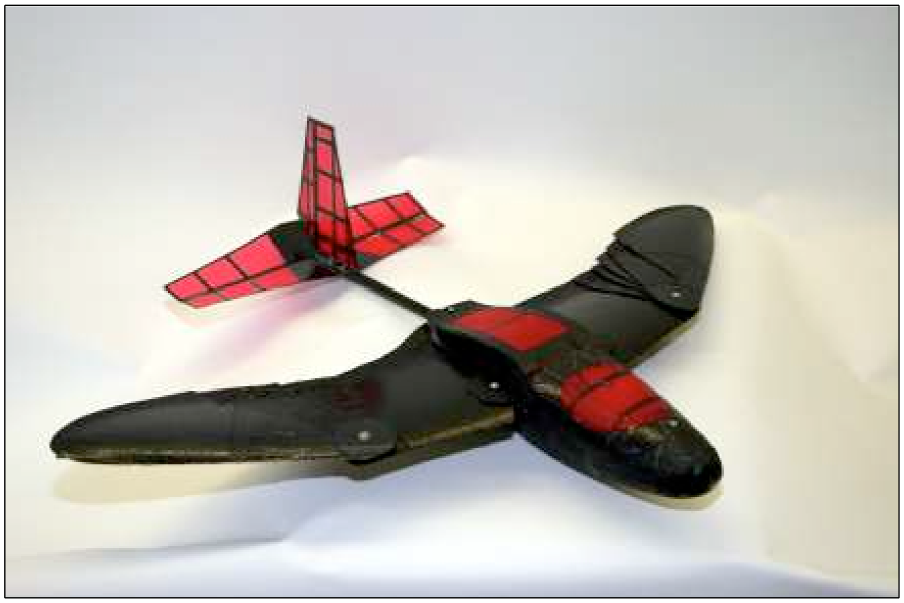


Figure 1-1. A micro air vehicle (MAV) from the University of Florida Flight Control Laboratory.

Such close-proximity flight presents challenges for path planning. In particular, the vehicle will need to aggressively maneuver among the dense obstacles to achieve the vantages that provide the required information as illustrated in Figure 1-2. Traditional approaches that choose trajectories based on kinematic models are suspect due to their reliance on an inner-loop controller to tightly track those trajectories. As such, the path planning should be restricted to consideration of trajectories that are inherently feasible according to the dynamic constraints of maneuvering.

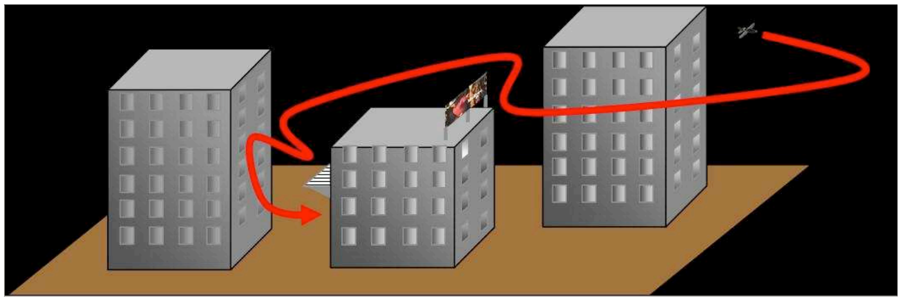


Figure 1-2. Micro air vehicle traveling through an urban environment.

Inclusion of dynamically-feasible motions in a planned trajectory is typically treated in either a direct or a decoupled fashion [1]. Direct planning methods, such as optimal control, consider a representation of the vehicle dynamics in the formulation of the planning problem and directly solve for optimal system inputs. Alternatively, indirect methods use a simplified model of vehicle motion to plan a reference path and then "smooth" the path to satisfy dynamics using methods such as feedback control. Direct methods compute optimal trajectories but are often intractable for realistic problem descriptions whereas indirect methods often exhibit tractable complexity properties that come at the expense of optimality.

Researchers have found clever ways to manipulate this tradeoff through a variety of techniques such that dynamics can be directly included in the planning process. For example, some researchers have recognized that systems which exhibit differential flatness properties admit solutions that can be represented parametrically in terms of a set of flat outputs and their derivatives [2–4]. Others have applied mixed-integer linear programming (MILP) to model dynamic constraints as a set of switching bounds on system velocities and accelerations [5–7]. Frazzoli et al introduced a planning technique that utilizes a "sampled-dynamics" model which employs a set of dynamically-consistent motion primitives [8, 9]. Additionally, recent advances in randomized planning allow the use of any of these techniques as local trajectory generation methods for growing a probabilistic tree of actions to explore the solution space [10–12].

The concept of motion primitives is a central theme for several of these investigations into feasible-path motion planning. A critical foundation was established by Dubins for a 2-dimensional (2-D) car [13] and has since been expanded into 3-D versions. One approach has been developed but does not deal with constraints in the climb rate or specific values of these climb rates as is the case with motion primitives [14]. A complete analogue to the Dubins car in 3-D is being developed to account for the shortest path between two points with associated heading constraints which decomposes the problem into different cases of which only some can be solved [15].

1.2 Problem Description

Given an initial location and heading in 3-dimensional space and the locations and dimensions of all the 3-dimensional obstacles in an environment, the trajectory to a goal location and heading is desired. The work presented in this thesis will address the problem of motion planning in 3-dimensional space. Specifically, motion planning for missions involving vehicles with 3-dimensional motion in close proximity to 3-dimensional obstacles is developed. The trajectory must be dynamically constrained to the performance attributes of the vehicle and must not intersect with any of the obstacles.

1.3 Approach Overview

The topic of 3-dimensional motion primitives is discussed in Chapter 2. Both 2-dimensional and 3-dimensional versions of the 2-primitive and 3-primitive motion sequences are discussed and developed respectively. The 3-primitive motion sequences are referred to as Dubins paths and are used in the development of the motion planner. Chapter 3 focuses on the subject of random dense trees (RDTs). These trees will be utilized for trajectory generation through the obstacle rich environment. The theory of randomized methods for path planning is discussed followed by the transition of 2-dimensional RDTs to 3-dimensional RDTs. In Chapter 4, the concept of 3-dimensional motion primitives and 3-dimensional RDTs are fused together into the motion planning algorithm. An explanation of how the algorithm functions is presented and discussed in detail. Chapter 5 utilizes this motion planning algorithm in four different examples to demonstrate the performance and characteristics of the algorithm.

1.4 Contribution

The implementation of the RDT as a trajectory generation tool in the motion planning algorithm is the contribution beyond previous work in the area of 3-dimensional motion planning [5, 14–17]. This produces a search that provides better optimality through environments with obstacles densely placed throughout; specifically, a condition to limit the branches between nodes is defined along with a terminal condition for tree growth. The branch limitation actually seeks to place more branches within the region to increase probability of finding a local minimum among the numerous solutions for paths through obstacles. The terminal condition notes that the exploration can change from a tree growth to a direct solution once an optimal path to the goal can be reached without intersecting any obstacles. In this way, the initial computation burden may slightly increase due to the extra nodes and additional terminal check; however, the resulting path will have lower cost.

CHAPTER 2
MOTION PRIMITIVES

2.1 Introduction

The concept of motion primitives leads to a useful framework for the simplification of complicated dynamic models. This framework involves combining sequences of compatible primitives to represent more complicated trajectories. A set of compatibility conditions are detailed in the literature [8]. This research utilizes path combinations, or motion primitives, consisting of straight paths, turning paths (both left and right), climbing paths, and diving paths. Motion primitive model theory is defined and described in more detail in the literature [18]. The progression of research from 2-dimensional (2-D) to 3-dimensional (3-D) motion primitive analysis is presented in the following sections.

2.2 Two-Dimensional Dubins Car

A standard model using motion primitives is known as the Dubins car [13, 19–23]. This simple 2-dimensional vehicle motion model operates in a configuration space, or $\mathcal{C}$-space, spanned by two Euclidean position variables, p_x and p_y, and an angle describing heading, ψ. Vehicle movement is restricted to driving straight or turning to either the left or right. The straight motion is constrained to a constant velocity while the turns are constrained to a constant velocity and turn rate. As such, the motion of the Dubins car is described by the differential system shown in Equation 2–1.

$$\begin{bmatrix} \dot{p}_x \\ \dot{p}_y \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} \cos\psi \\ \sin\psi \\ \omega \end{bmatrix} \tag{2–1}$$

The discrete set of values assumed by the turn rate ω is shown by Equation 2–2.

$$\omega \in \{-1, 0, +1\} \ /sec \tag{2–2}$$

The Dubins car is an especially interesting model in that a closed-form solution for optimal trajectories has been derived [13]. This solution notes the path from any initial point (x_o, y_o) and heading (ψ_o) to any final point (x_f, y_f) and heading (ψ_f) can be expressed using only the proper sequence of only 3 primitives. There are 6 combinations of these 3 primitive sequences and are divided into Turn-Straight-Turn and Turn-Turn-Turn categories, as detailed in Table 2-1. Such a result allows strategies using optimal control to be directly compared with a global minimum for evaluating techniques of path planning.

Table 2-1. Dubins car primitive sequences.

Turn-Straight-Turn	Turn-Turn-Turn
Left-Straight-Left	Right-Left-Right
Right-Straight-Right	Left-Right-Left
Right-Straight-Left	
Left-Straight-Right	

The transformations describing the left turn, right turn, and straight ahead motions are shown as Equations 2–3, 2–4, and 2–5, respectively. The closed-form expressions for the trim durations τ are presented in the literature [18, 24].

$$L(p_x, p_y, \psi, \tau) = (p_x + \sin(\psi + \tau) - \sin\psi,\ p_y - \cos(\psi + \tau) + \cos\psi,\ \psi + \tau) \tag{2–3}$$

$$R(p_x, p_y, \psi, \tau) = (p_x - \sin(\psi - \tau) + \sin\psi,\ p_y + \cos(\psi - \tau) - \cos\psi,\ \psi - \tau) \tag{2–4}$$

$$S(p_x, p_y, \psi, \tau) = (p_x + \tau\cos\psi,\ y + \tau\sin\psi,\ \psi) \tag{2–5}$$

Examples of solutions from an initial position and heading to a final position and heading are shown in Figure 2-1 for a Dubins car. All six Dubins car combinations are utilized and plotted. The optimal solution, which corresponds to the lowest travel time, is highlighted.

In addition to the 3-primitive sequences listed above, there is also a family of 2-primitive sequences utilizing only the Turn-Straight methodology (Left-Straight and Right-Straight). This combination will provide a path from any initial point (x_o, y_o) and heading (ψ_o) to any final point (x_f, y_f) but the final heading (ψ_f) cannot be guaranteed

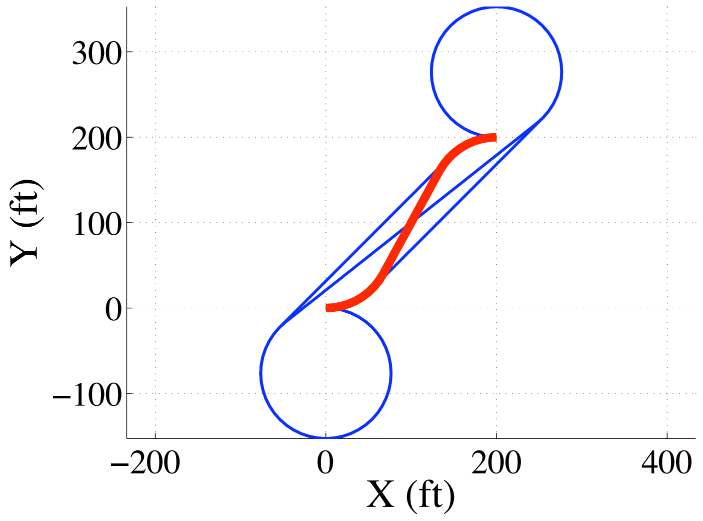


Figure 2-1. Possible (—) and optimal (▬) 2-D Dubins path solutions.

using the 2-primitive sequence in view of the fact that a third primitive is necessary to provide the desired heading. Examples of solutions from an initial position and heading to a final position are shown in Figure 2-2 for a 2-primitive Turn-Straight solution sequence. There are 3 examples of both the Left-Straight and Right-Straight paths, each with a different turn rate ω.

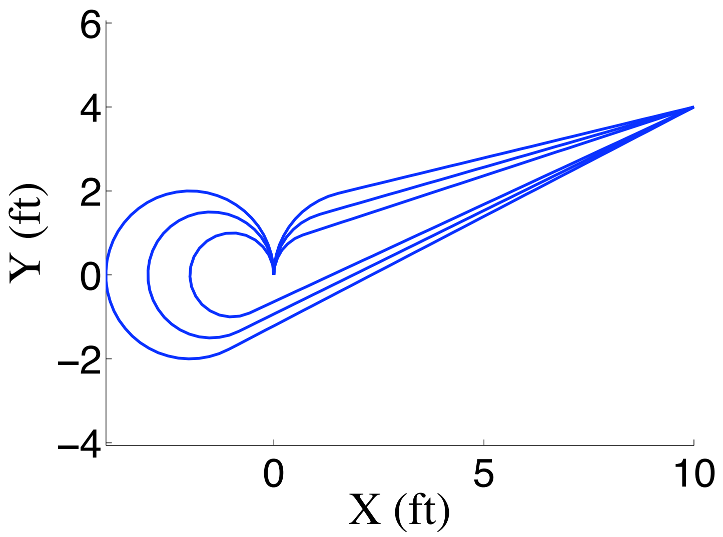


Figure 2-2. Turn-straight solution sequences.

2.3 Three-Dimensional Dubins Airplane

Utilizing the theory of 2-dimensional motion primitives and the Dubins car, a 3-dimensional version of motion primitives is developed that creates an aircraft version of the Dubins car. This model operates in a $\mathcal{C}$-space spanned by three Euclidean position variables, p_x, p_y, and p_z, and an angle describing heading, ψ. Such a development adds a constant rate of change in altitude to the constant rate of change of turn already used by the 2-D Dubins car. The resulting dynamics are described in Equation 2–6.

$$\begin{bmatrix} \dot{p}_x \\ \dot{p}_y \\ \dot{p}_z \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} \cos\psi \\ \sin\psi \\ \gamma \\ \omega \end{bmatrix} \tag{2–6}$$

Discrete values of turn rate, ω, and climb rate, γ, are defined in Equations 2–7 and 2–8.

$$\omega \in \{-1, 0, +1\} \;\; /sec \tag{2–7}$$

$$\gamma \in \{-1, 0, +1\} \;\; ft/sec \tag{2–8}$$

This solution notes the path from any initial point (x_o, y_o, z_o) and heading (ψ_o) to any final point (x_f, y_f, z_f) and heading (ψ_f) can be expressed using only the proper sequence of only 3 primitives. There are 18 combinations of these 3 primitive sequences organized into 3 groups of 6 combinations. These are detailed in Tables 2-2, 2-3, and 2-4.

Table 2-2. Dubins airplane climbing primitive sequences.

Turn-Straight-Turn while Climbing	Turn-Turn-Turn while Climbing
(Left-Straight-Left)Climb	(Right-Left-Right)Climb
(Right-Straight-Right)Climb	(Left-Right-Left)Climb
(Right-Straight-Left)Climb	
(Left-Straight-Right)Climb	

Table 2-3. Dubins airplane level primitive sequences.

Turn-Straight-Turn with Level Flight	Turn-Turn-Turn with Level Flight
(Left-Straight-Left)Level	(Right-Left-Right)Level
(Right-Straight-Right)Level	(Left-Right-Left)Level
(Right-Straight-Left)Level	
(Left-Straight-Right)Level	

Table 2-4. Dubins airplane diving primitive sequences.

Turn-Straight-Turn while Diving	Turn-Turn-Turn while Diving
(Left-Straight-Left)Dive	(Right-Left-Right)Dive
(Right-Straight-Right)Dive	(Left-Right-Left)Dive
(Right-Straight-Left)Dive	
(Left-Straight-Right)Dive	

An optimal path can be computed as a sequence of these motion primitives to traverse from an initial position and heading to a final position and heading. An example scenario is shown in Figure 2-3 that demonstrates all of the Turn-Straight-Turn primitive sequences along with the optimal path for climbing, level flight, and diving cases.

It must be noted that there are scenarios where the plane cannot reach the goal position. The vehicle model has a prescribed inability to travel directly vertical or horizontal and thus cannot reach a set of final conditions due to limitations on climb rate and turn rate. Pachikara has studied this problem in his research and implementation of that work can be used to address this issue [25]. For all of the research presented here, the initial point and goal point are sufficiently separated in the XY-plane such that the vehicle's turn rate and/or climb rate do not prevent the vehicle from attaining the goal configuration.

In addition to the 3-primitive Dubins airplane sequences listed above, there is also a family of 2-primitive sequences utilizing only the Turn-Straight methodology (Left-Straight and Right-Straight) just as in the 2-dimensional case but including the constant rate of change in altitude. This combination will provide a path from any initial point (x_o, y_o, z_o) and heading (ψ_o) to any final point (x_f, y_f, z_f) but the final heading (ψ_f) cannot be

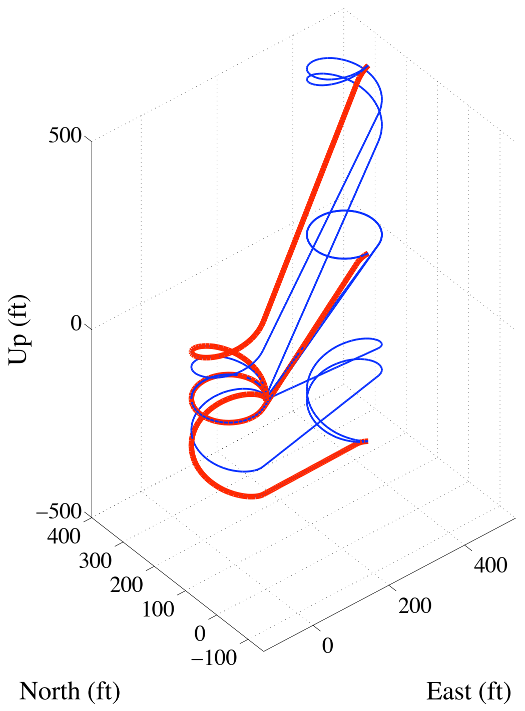


Figure 2-3. Possible (—) and optimal (▬) 3-D Dubins airplane solutions.

guaranteed using the 2-primitive sequence owing to the fact that a third primitive is necessary to provide the desired heading. These primitives are graphically shown in Figure 2-4 as 9 possibilities. The horizontal motion has only 3 possibilities of going left-straight, right-straight, and straight only ($\psi = 0$). These horizontal motions are coupled with independent vertical motion that allows level flight and climb or dive.

2.4 Library

The motion primitives represent maneuvers that the vehicle can perform. Since most vehicles can vary their rates of change, it is reasonable to define a set of parameters such that $\Omega = \{\omega_1, ..., \omega_n\}$ represents the set of possible turn rates and $\Gamma = \{\gamma_1, ...\gamma_m\}$ represents the set of possible climb rates.

A set of motion primitives can then be defined that represent all possible maneuvers. Each element in this set is actually a trajectory defined by the time-varying values of

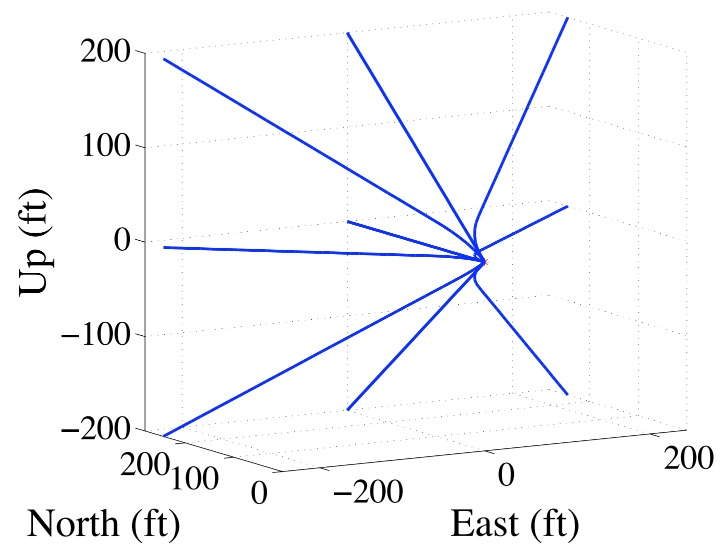


Figure 2-4. Turn-straight solution sequences for 3-D.

position and orientation during the maneuver. As such, each element is parametrized by the duration of the maneuver, τ, turn rate, ω, climb rate, γ, to result in set of $\mathcal{X}$.

$$\mathcal{X} = \left\{ X(\tau,\omega,\gamma) \quad : \quad X = \begin{bmatrix} p_x(0:\tau) \\ p_y(0:\tau) \\ p_z(0:\tau) \\ \psi(0:\tau) \end{bmatrix} \in Eq\ 2-6, \omega \in \Omega, \gamma \in \Gamma \right\} \tag{2–9}$$

A critical feature of this set is the notion of feasibility. Essentially, any member, $X \in \mathcal{X}$, is constrained so the evolution of the trajectory must follow the dynamics of Equation 2–6. The resulting set is a collection, or library, of feasible maneuvers that can be achieved by the vehicle.

The configurations which may be reached by the vehicle after a single maneuver can thus be expressed using this parameterized set. An initial configuration of position and orientation can be defined as $C(0) \in \mathcal{R}^4$ and the ensuing configuration at any time, $t \in \mathcal{R}$, can be defined as $C(t) \in \mathcal{R}^4$ as a result of initiating some maneuver, $X(t, \omega, \gamma)$ as in Equation 2–10.

$$C(t) = C(0) + X(t, \omega, \gamma) \tag{2–10}$$

CHAPTER 3
RANDOM DENSE TREES

3.1 Introduction

Randomized methods for path planning have been formulated to consider systems with complicated dynamics. The fundamental feature of such methods is a localized approach that considers sequentially expanding into a search space to rapidly and efficiently find sub-optimal solutions. A variety of methods, including probabilistic roadmaps and random dense trees (RDTs), have been developed; however, the use of random dense trees will be adopted due to its ability to directly handle motion primitives and generate feasible trajectories for models of realistic vehicles [26–31].

3.2 Two-Dimensional Random Dense Trees

The material in this section is taken directly from Section 3.3 of *Trajectory Generation for Effective Sensing of a Close Proximity Environment* by Kehoe [18].

Random dense tree (RDT) based planners provide an alternative to the basic probabilistic roadmap method paradigm that enables efficient solutions to differentially constrained problems. While the probabilistic roadmap method generates a roadmap that describes the connectivity of many configurations to many other configurations, RDT methods generate a tree that is rooted at a specific initial condition and which describes connectivity of this initial condition to as many reachable configurations as possible. Algorithmic details ensure efficient and rapid exploration of the space. A drawback of RDT methods is that they are designed to solve a single planning problem at a time. This limitation is in contrast to probabilistic roadmap method planning algorithms, which establish a network that spans the configuration space, or $\mathcal{C}$-space, and can be used many times for many different planning tasks. A major benefit in this tradeoff is that RDT methods can often handle problems involving dynamic systems. In general, RDT methods incrementally build a search tree from an initial node in three main steps:

1. **Node Selection:** A node from the existing tree is selected as a location to add a branch. Selection of a particular node is usually based on probabilistic criteria that may require use of a valid distance metric.

2. **Node Expansion:** A local planning method is used to extend a feasible trajectory from the selected node. The local goal for this trajectory branch is determined probabilistically.

3. **Evaluation:** The new branch is evaluated according to performance criteria and often for connection to the goal configuration. Additionally, the new branch may be subdivided into multiple segments, thus adding several new nodes to the existing tree.

A variety of RDT-based planners have been developed with numerous variations on the main steps listed previously, often to optimize performance for a specific application or to address a pathological case [26–31]. Two algorithms, the Rapidly-exploring Random Tree (RRT) algorithm and the Expansive Spaces Tree (EST) algorithms, demonstrate different core exploration philosophies through the manner in which nodes are selected and expanded. These algorithms also serve as a basis for many of the existing variations on the general method, and hence prove useful as demonstrative examples.

3.2.1 Rapidly-Exploring Random Trees (RRT)

The RRT algorithm was developed by Lavalle and Kuffner specifically to handle problems that involve dynamics and differential constraints [12, 28]. The algorithm biases tree growth toward unexplored areas of the space and hence focuses on rapid exploration.

The node selection step is initiated with a sampled configuration that is chosen from a uniform distribution of the configuration space, or $\mathcal{C}$-space. A distance metric is then used to determine the closest point in the existing tree. During the expansion step, the selected node is extended incrementally "toward" the sampled configuration using a local planning method. This incremental extension is performed to varying degrees in different versions of the algorithm and is ultimately a design parameter. Some versions use a fixed step size, others use a step size proportional to the distance from the sample, while others attempt to completely connect the sampled configuration to the existing tree.

Figures 3-1A and 3-1B depict the RRT expansion process. Both images show a tree grown from the root node, N_0, in a two-dimensional $\mathcal{C}$-space that contains obstacles. Figure 3-1A depicts the sampling step, in which a random configuration, N_{rand}, is selected and the nearest node in the existing tree, N_{near}, is determined. Figure 3-1B shows the expansion step, where a branch is incrementally extended from N_{near} toward N_{rand} along the trajectory connecting the two configurations. A new node, N_{new}, is added at the end of the new branch. The algorithm proceeds in this fashion until a branch of the tree reaches the goal within some specified tolerance.

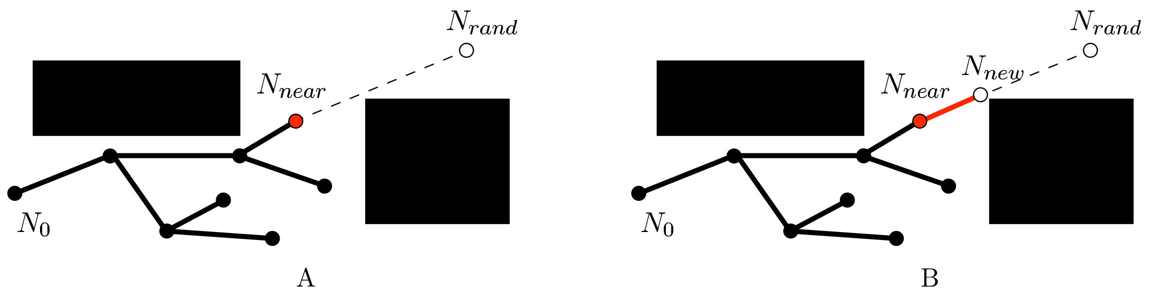


Figure 3-1. RRT algorithm. A) Sampling step. B) Expansion step.

3.2.2 Expansive-Spaces Trees (EST)

The EST algorithm was developed by Hsu et al as a planning method to address problems involving high-dimensional $\mathcal{C}$-spaces and was later adopted to handle kinodynamic planning problems [10, 32]. The EST algorithm explores space in a fundamentally different way than the RRT algorithm. Specifically, node selection occurs through the random selection of an existing node according to a probability distribution that is left as a design choice. This node is expanded within a local neighborhood that is defined by a valid distance metric. A configuration is sampled randomly from within this neighborhood and a local planning method is used to connect the selected node to the sampled configuration.

Figures 3-2A and 3-2B depict the EST expansion process. Both images show a tree grown from the root node, N_0, in a two-dimensional $\mathcal{C}$-space that contains obstacles. Figure 3-2A depicts the node selection step, in which the expansion node, N_{exp}, is selected from the existing nodes. The neighborhood of N_{exp} is defined here using a Euclidean distance metric and is shown as the area within the dashed circle in Figure 3-2A. Figure 3-2B shows the expansion step, where a random configuration, N_{rand} is selected from the neighborhood of N_{exp} and then a trajectory is planned from N_{exp} to N_{rand}. The algorithm proceeds in this fashion until a branch of the tree reaches the goal within some specified tolerance.

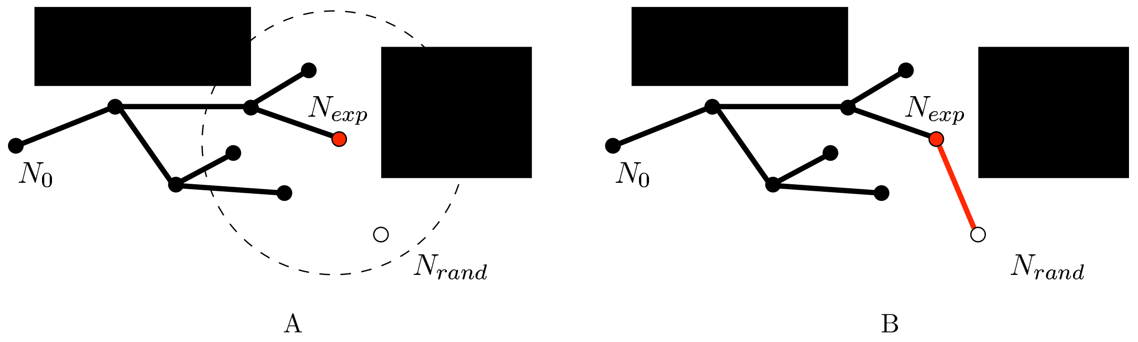


Figure 3-2. EST algorithm. A) Node selection. B) Sampling and expansion.

3.2.3 Discussion

It is important to note the fundamental difference between the ways in which the RRT and EST explore the space. Samples from empty space have a tendency to "pull" branches off of the tree built in the RRT algorithm. Thus, the space is rapidly spanned with coarse resolution. Continued sampling has the effect of improving the resolution of this exploration without appreciably changing the form of the solution. This concept is depicted in Figure 3-3A. Conversely, the EST selects a node randomly and tends to "push" branches from the selected node toward empty space as shown in Figure 3-3B. A benefit to this "pushing" tendency is that the shape of the tree is continually evolving such that expansion is guided by the node sampling distribution. A wise choice of this distribution can favorably affect solution performance qualities; however, care must be taken to avoid biasing exploration toward previously explored areas.

3.3 Three-Dimensional Random Dense Trees

The concept of random trees is also useful for considering a 3-dimensional (3-D) Dubins airplane. The basic tree structure actually allows exploration into any n-dimensional

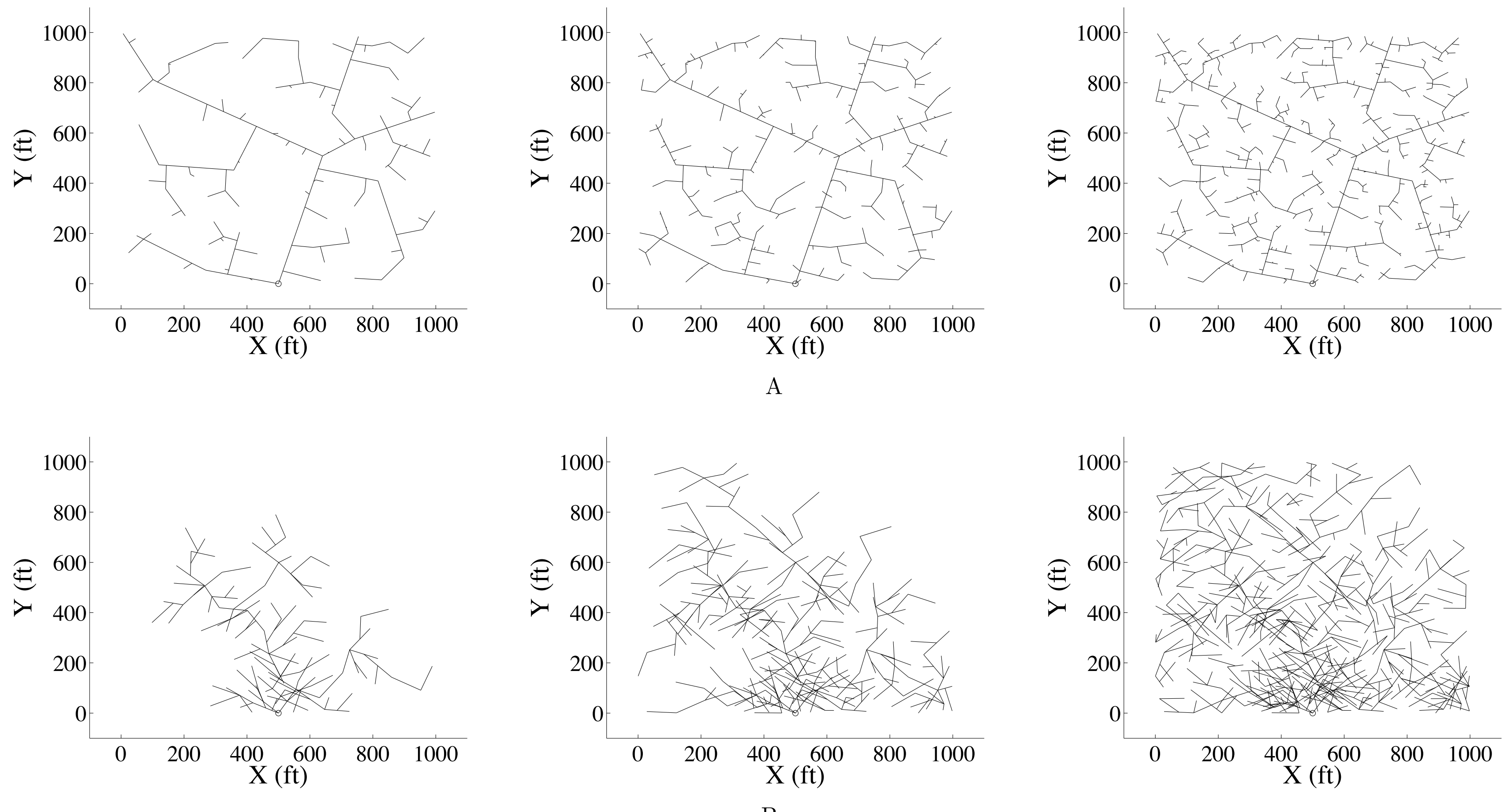


Figure 3-3. Differences in exploration strategy for the RRT algorithm vs. the EST algorithm. A) RRT expansion. B) EST expansion.

space so a 3-dimensional tree is a relatively straightforward extension of the 2-dimensional (2-D) tree. As with the 2-dimensional case, both the RRT and EST approaches can be modified to be implemented into 3-dimensions. For the remainder of this investigation, the pull expansion philosophy exhibited by the RRT algorithm will be utilized. Subsequent to experimentation with the two expansion philosophies, the RRT approach is selected rather than the EST approach due to its superior efficiency, computation time, and robustness attributes in finding feasible solutions.

The growth of the tree follows a standard algorithm using nodes and branches. A set of nodes are randomly placed into the environment within some limit on range. An example is shown in Figure 3-4 to demonstrate a 3-dimensional RRT at iteration counts of 100, 250, and 500.

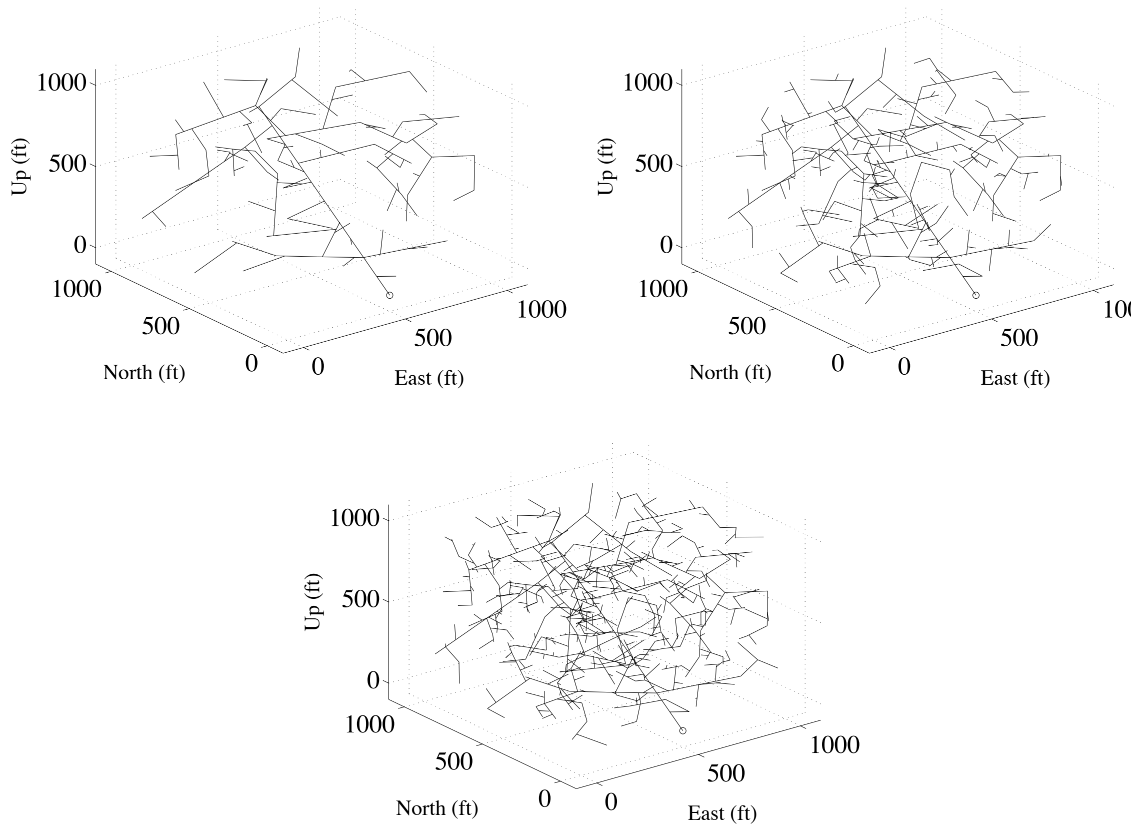


Figure 3-4. Growth of 3-D Tree after 100 Iterations (top left), 250 Iterations (top right) and 500 Iterations (bottom)

3.4 Three-Dimensional Random Dense Tree Trajectory Generation

The nature of the exploration and the associated termination conditions of the 3-dimensional RDT developed in Section 3.3 are issues that still need to be addressed. The combination of this 3-D RDT and the dynamic motion primitives described in Chapter 2 are the basis for the development of a trajectory generation algorithm. To combine the ideas, the nodes for the tree growth are randomly placed into the environment within some limit on range placed by probabilistic assessment of the vehicle velocity. The validity of each node is then assessed by determining if a sequence of turn maneuver, X_t, and straight maneuver, X_s, could reach that location, $C(t+\tau)$, from the previous node location, $C(t)$. In other words, the tree's branches are created by a series of Turn-Straight maneuvers like those detailed in Section 2.3. Several of these Turn-Straight sequences may reach the required new node so the final path, or branch, is selected by minimizing the time required for travel as shown in Equation 3–1.

$$\min_{\substack{X_t \in \mathcal{X} \\ X_s \in \mathcal{X}}} \quad \tau_1 + \tau_2 \tag{3–1}$$

subject to

$$C(t+\tau_1+\tau_2) = C(t) + X_t(\tau_1, \omega, \gamma) + X_s(\tau_2, 0, \gamma)$$

A representative example is shown in Figure 3-5 to demonstrate a tree built upon the motion primitives in $\mathcal{X}$. Each branch consists of a turn motion followed by straight motion with each motion obtained as a feasible maneuver from the library. The total length of each maneuver is jointly determined by the minimization of Equation 3–1.

Path planning into 3-dimensional space using trees can be constrained to account for obstacles. Obviously any resulting path must be feasible both in the sense that the

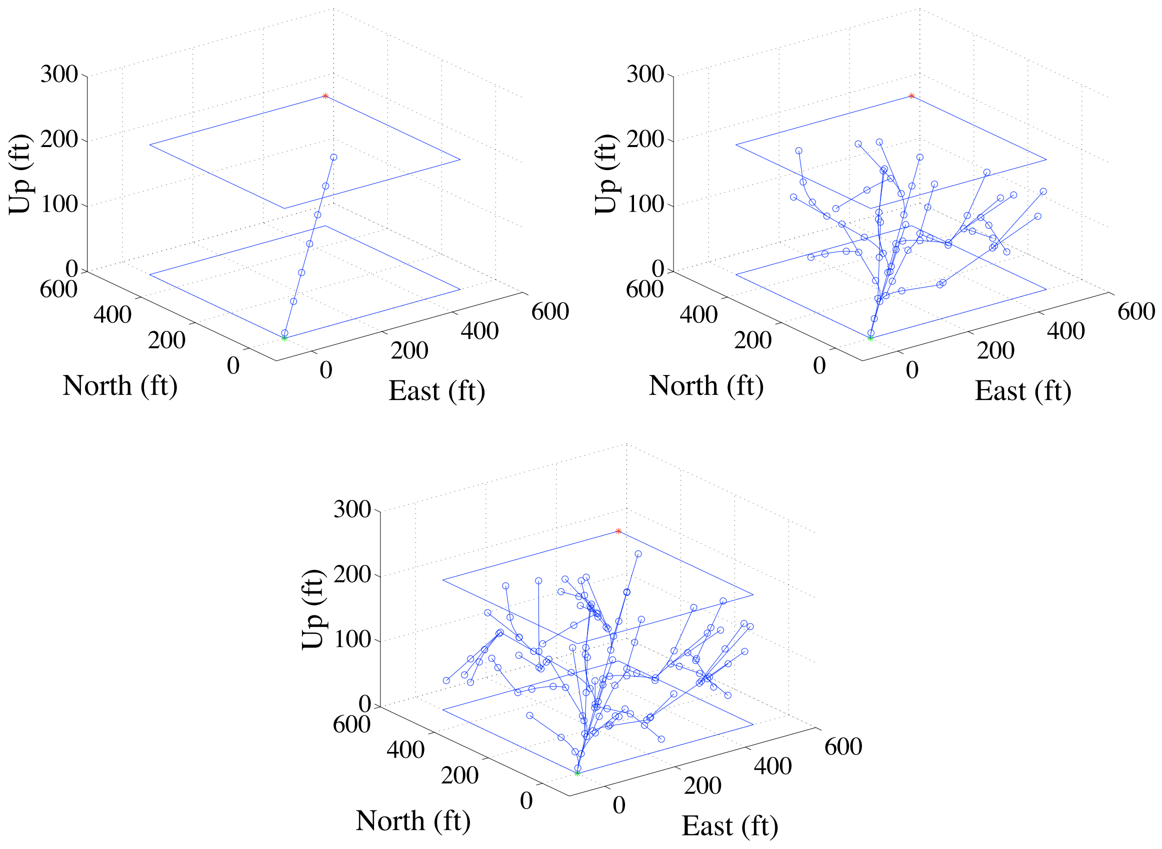


Figure 3-5. Growth of 3-D dynamically constrained random dense tree into unoccupied space.

vehicle can follow the trajectory and in the sense that the trajectory does not intersect any obstacles.

A pruning method is used to ensure obstacle avoidance. This method does not directly consider the location of the obstacles to optimize tree growth; rather, it simply prunes nodes and branches that lie within an obstacle. The node selection thus remains random with some of the nodes being eliminated by a check on the node location and the obstacle locations.

A representative expansion into a 3-D space with obstacles is shown in Figure 3-6. The tree grows and branches are formed along paths that do not intersect any obstacles due to the pruning approach. The resulting path is thus able to avoid obstacles and reach the goal configuration.

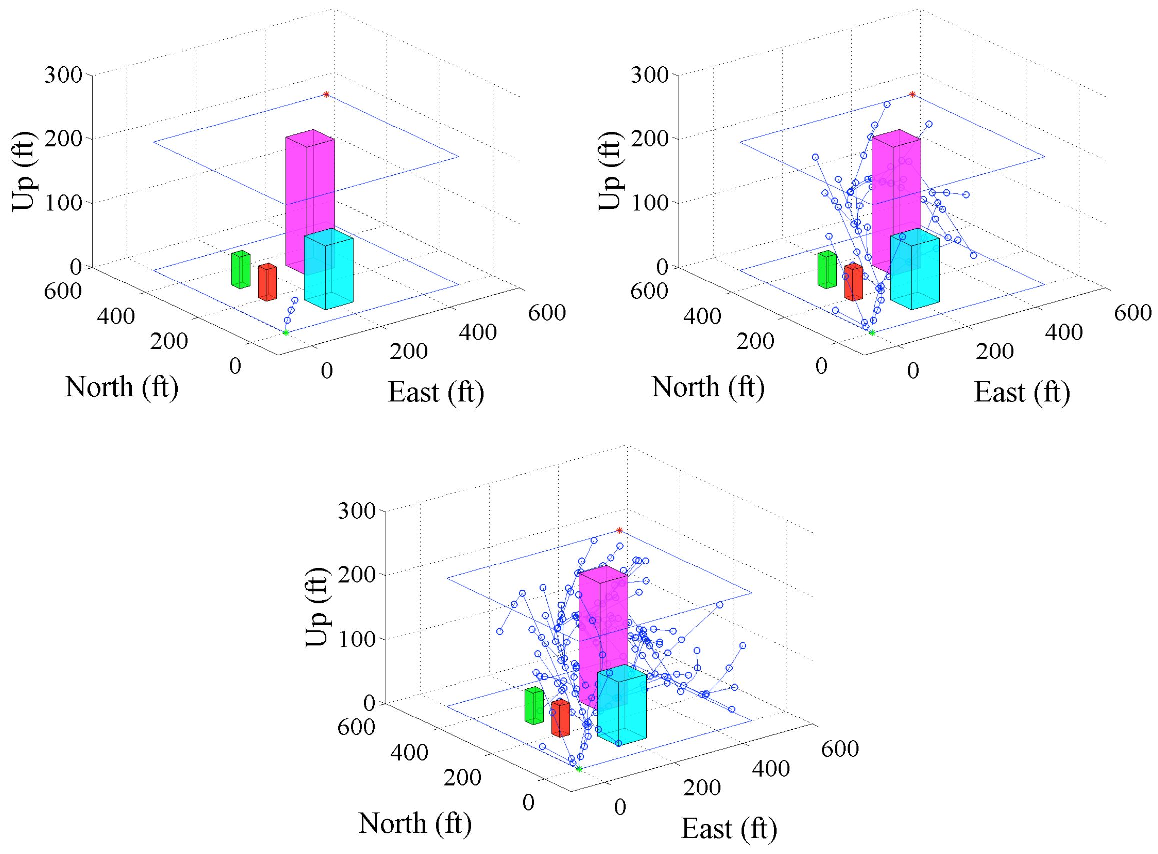


Figure 3-6. Growth of 3-D dynamically constrained random dense tree into occupied space.

This pruning notes that a set of locations, $\mathcal{O}$, may be defined that encompasses the obstacles. The definition in Equation 3–2 uses a simple orthogonal polyhedron approximation such that each obstacle has limits on east range, $[x_1, x_2]$, north range, $[y_1, y_2]$, and altitude range, $[z_1, z_2]$, for k obstacles.

$$\mathcal{O} = \left\{ O \quad : \quad O = \begin{bmatrix} x \in [x_1^i, x_2^i] \\ y \in [y_1^i, y_2^i] \\ z \in [z_1^i, z_2^i] \end{bmatrix} \forall i \in [1, k] \right\} \tag{3–2}$$

The growth of the tree occurs such that a node, $C(t+\tau_1+\tau_2)$, as in Equation 3–1 is valid if neither that node nor a path to that node intersect any obstacles as described in Equation 3–3:

$$\begin{gathered} C(t+\tau_1+\tau_2) \notin \mathcal{O} \\ \exists X \in \mathcal{X} \;\; with \;\; X \notin \mathcal{O} \end{gathered} \tag{3–3}$$

CHAPTER 4
MOTION PLANNING

4.1 Introduction

Motion planning describes the process of developing the transformation of a system from an initial configuration to a terminal or goal configuration. System configurations are defined as a vector, $\vec{x}_c$, on the configuration space, or $\mathcal{C}$-space. The $\mathcal{C}$-space encompasses the variables that describe the position and orientation of the body coordinate frame including the terminal or goal configuration.

Motion planning problems for vehicle systems are typically classified according to three main categories [33]: point-to-point motion, path following, and trajectory tracking. In all of these cases, the vehicle is required to move from an initial configuration to a goal configuration. The differences occur in how the function describing the vehicle's motion is constrained. In the case of point-to-point motion, there are no restrictions on the transitional motions occurring between the beginning and final configurations but just that final configuration is reached. Point-to-point motion planning is solved via a series of waypoints. For the path-following case, the vehicle is instructed to obey a continuous path in the $\mathcal{C}$-space prescribed by system differential constraints and which has the initial and goal configurations as endpoints. The plans for path following motion are developed by way of a function defined on the $\mathcal{C}$-space. The trajectory tracking case is identical to the path-following case with the addition of a time requirement. Trajectory motion plans are defined as either a system input function, $\vec{u}$, that is a function of time or as time-parameterized functions defined on the $\mathcal{C}$-space.

In this section, an algorithm for motion planning is formulated for generating guidance trajectories for a vehicle system given a model of the vehicle motion and a known environment. The environment in which the vehicle functions is occupied with densely placed 3-dimensional (3-D) obstacles that force the vehicle to travel within close proximity to the obstacles. The vehicle's path is generated using 2 parts. The first portion combines the 3-D motion primitives with the 3-D rapidly-exploring random tree. This segment of the path will travel amongst the obstacles. The second portion of the path implements the 3-D Dubins path to travel from the end of a tree branch to the goal point. This combination will produce approximate minimum-time trajectories for the constrained system.

4.2 Model

The motion planner developed utilizes a motion primitive model that behaves according to the dynamics described by Equation 4–1, which is an extension of the Dubins airplane model examined in Section 2.3. The constant translational velocity, V, is restricted differentially to act forward in the direction of the vehicle heading. The motion of the of the vehicle is controlled via the inputs to the differential system: the turn rate, ω, and the climb rate, γ. The system described by Equation 4–1 admits trim trajectories that belong to three families: constant rate turns (left and right), constant rate climbs and dives, and straight forward motion.

$$\begin{bmatrix} \dot{p}_x \\ \dot{p}_y \\ \dot{p}_z \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} \mathrm{V}cos\psi \\ \mathrm{V}sin\psi \\ \gamma \\ \omega \end{bmatrix} \tag{4–1}$$

Here, trim trajectories behave according to the kinematic conditions shown in Equations 4–2, 4–3, and 4–4. A motion primitive model can be formed by selecting a set of trim primitives that behave according to these conditions.

$$V = \text{const.} \tag{4–2}$$

$$\dot{p}_z = \gamma = \text{const.} \tag{4–3}$$

$$\dot{\psi} = \omega = \text{const.} \tag{4–4}$$

There are a total of $3m(2n + 1)$ trim primitives selected. These trim primitives consist of constant-rate turns and constant-rate climbs at n different turn rates and m different climb rates in each direction. In addition, there is a straight-ahead primitive that corresponds to $\omega = 0$ and $\gamma = 0$, purely climbing and purely diving primitives corresponding to $\omega = 0$, and purely turning (both left and right) primitives corresponding to $\gamma = 0$. This set is shown as Equations 4–5 and 4–6. The velocity, V, is held fixed over the set of all primitives.

$$\dot{\psi} \in \{0, \pm\omega_1, \pm\omega_2, \cdots, \pm\omega_n\} \qquad |\omega_i| \leq \dot{\psi}_{max}, i = 1, \cdots, n \tag{4–5}$$

$$\dot{p}_z \in \{0, \pm\gamma_1, \pm\gamma_2, \cdots, \pm\gamma_m\} \qquad |\gamma_i| \leq \dot{p}_z, max, i = 1, \cdots, m \tag{4–6}$$

4.3 Overview

The path creation algorithm is executed using a series of primary steps. A detailed explanation of these steps are located in the literature [18] but are summarized as follows:

4.3.1 Select a Node

A point is selected from the subspace of the feasibility space which is spanned by the position variables. An approximate, obstacle-free distance metric is used to determine the nearest node in the existing tree.

4.3.2 Extend a Branch

The set of $2n$ unique solutions on the position subspace are enumerated, evaluated, and pruned. Selection criteria is used to choose a branch from the set for addition to the solution tree.

4.3.3 Check for Solutions

The new branch is split into an intermediate set of nodes. At each node, a check is presented for obstacle-free connection to the goal configuration on the full $\mathcal{C}$-space using the optimal-control solution. When a lower cost solution is found, the new solution is added to a solution list and the current upper bound is updated. This obstacle-free path check notes if a direct solution from the current node, $C(t)$, to the goal location, C_g, exists for the standard 3-sequence Dubins solution. This check can be described as:

```
if $\exists X_1, X_2, X_3$ such that
    $C_g = C(t) + X_1(\tau_1, \omega_1, \gamma_1) + X_2(\tau_2, \omega_2, \gamma_2) + X_3(\tau_3, \omega_3, \gamma_3)$
    then stop tree growth
else choose new $C(t + \tau_1 + \tau_2)$ as in Equation 3–1
end
```

4.4 Discussion

It should be emphasized that the tree growth uses a sequence of two maneuvers while the final direct solution uses a sequence of three maneuvers. This discrepancy notes that a 3-sequence path is optimal for a Dubins vehicle but a 2-sequence path is preferable for maneuvering through obstacles. Essentially, the 2-sequence path will limit the range between nodes and thus force the nodes to lie closer together. Such close nodes will add more nodes to the final solution and, while incurring a small computational cost, will result in more sub-optimal solutions for the resulting path [16]. A local minimum with lower cost is anticipated using this algorithm since more sub-optimal solutions are computed. It is anticipated that fewer nodes will be more efficient for less-obstacle-dense environments but more nodes will be more efficient for more-obstacle-dense environments.

Also, to clarify the recording function of the algorithm, the process above is modified to improve upon the optimality of the solution. While the tree growth iterations are completed, path solutions will be found. After a solution (path) is computed, the solution's associated cost function (in this case the overall travel time) is compared to any previous solution costs. When a lower cost solution (smaller total travel time) is identified, the solution is recorded as the "most favorable solution". The tree is prescribed to continue to expand a fixed number of iterations and additional solutions will be found and checked in the same fashion. After all the iterations are complete, the final “most favorable solution” will be presented as the best sub-optimal solution.

CHAPTER 5
EXAMPLES

5.1 Introduction

Several examples of motion planning for an aircraft through a variety of environments are presented in this chapter. A set of initial conditions and final conditions are chosen for each example to reflect a mission of traversing the regions. In the first case, an obstacle-free environment is investigated and a path is created using only a 3-dimensional (3-D) Dubins Airplane path. In the second and third examples, the full motion planning algorithm is utilized for an urban environment with a modification to the random dense tree (RDT) growth algorithm being presented in the latter example. In the fourth and final example, the full motion planning algorithm is utilized once again but for an urban environment that includes an elevated bridge and a covered walkway. The dynamic properties for the formulation of the motion primitives are listed in Table 5-1 and are based on measurements from a class of micro air vehicles from the Flight Control Laboratory at the University of Florida. These vehicle properties will be used for all of the examples in this chapter.

Table 5-1. Vehicle properties for examples.

Property	Value
forward velocity	40 ft/s
turn radius	76 ft
max climb rate	30 ft/s

5.2 Three-Dimensional Dubins Airplane Paths (No Obstacles)

In this example, an obstacle free environment is assumed. Since there are no obstacles that need to be avoided, the tree portion of the motion planning algorithm is not necessary and only the second portion of the motion planning algorithm, the 3-dimensional Dubins airplane sequence, is implemented.

The vehicle starts at a position of (0,0,0) and a heading of 90^o (due East) while it is required to end at a position of (400,400,100) and a heading of 90^o. The results of the simulation are presented in Figure 5-1. There are 3 paths presented, each with a lower cost function than the one previous. The cost function to be minimized is defined to be purely the time to traverse the region. The first path created will be identified as the first "best" solution. As the simulation runs through all of the possible Dubins airplane possibilities, the simulation defines a path as being the "best" if it has a smaller overall travel time than the "best" path created before it. Recalling from Table 2-2 that there are six total combinations of Dubins airplane paths for climbing, only three are produced here. The other three are not presented because as they were utilized in the algorithm, their travel time was not smaller than the previously defined "best" path. The algorithm decided not to produce the path and instead move to the next possible Dubins airplane motion primitive combination listed in the algorithm. In this example, the (Right-Straight-Right)Climb sequence path is defined first as having a travel time of 36.9 seconds. Proceeding through the simulation, the (Left-Straight-Left)Climb sequence path is defined as having a shorter path travel time of 27.9 seconds. Finally, the (Left-Straight-Right)Climb sequence path is defined as having the shortest path travel time of all at 16.4 seconds. Inspection of the plots clearly shows that the climbing Left-Straight-Right sequence has the lowest cost function of all as it is almost a straight path from the initial configuration to the goal configuration.

5.3 Full Motion Planning in Urban Environment

In this example, the full motion planning algorithm is utilized for an urban environment. The vehicle starts at a position of (0,0,0) and a heading of 30^o while it is required to end at a position of (500,500,200) and a heading of 90^o. There are 4 obstacles, or buildings, of various sizes that lie throughout the region and must be strictly avoided. The details of the obstacles are presented in Table 5-2.

The tree portion of the motion planning algorithm is utilized for maneuvering through the obstacle rich portion of the environment, followed by the 3-dimensional Dubins airplane sequence for traveling from the end of the branches to the goal point and heading.

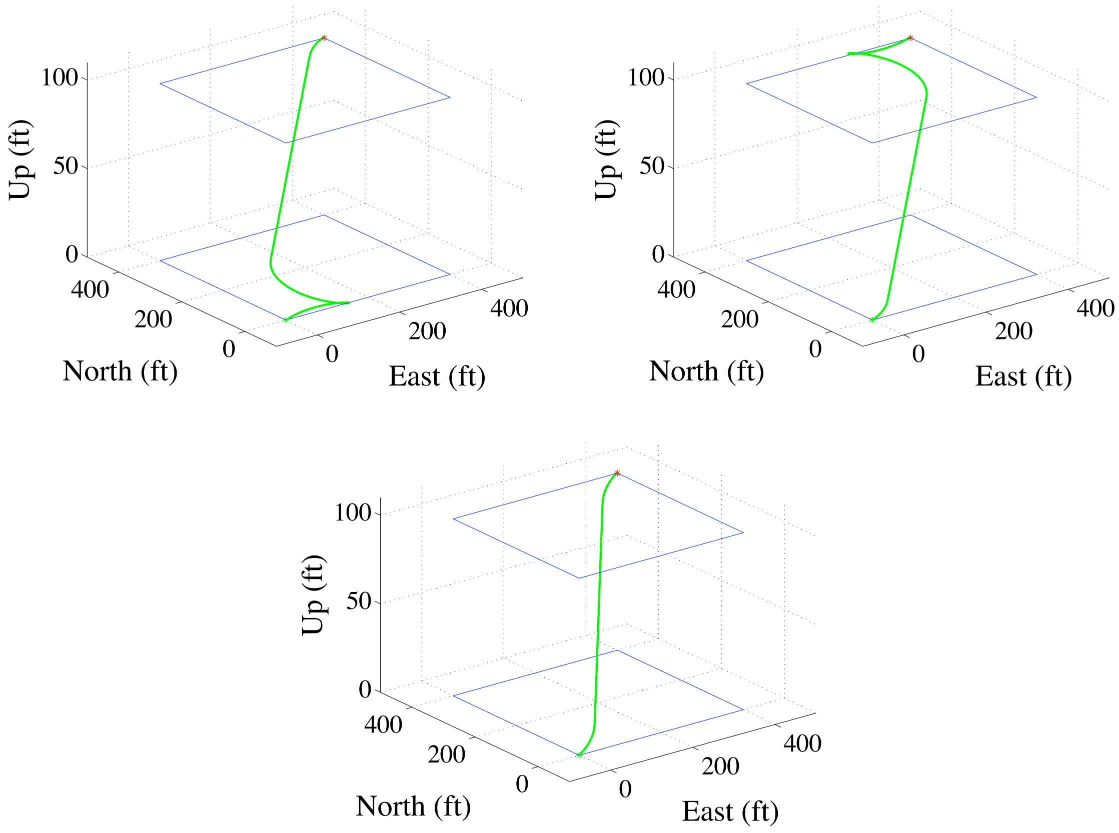


Figure 5-1. 3-D Dubins airplane paths using RSR (top left), LSL (top right), and LSR (bottom) motion primitive sequences. RDT portion of motion planning is not needed because there are no obstacles.

Table 5-2. 3-D obstacle dimensions and locations for urban environment. All units in feet.

Obstacle	Coordinates of Center	dx	dy	dz
Cyan	(200,100,50)	240-160=80	140-60=80	100-0=100
Red	(100,200,25)	115-85=30	215-185=30	50-0=50
Green	(100,300,25)	115-85=30	315-285=30	50-0=50
Pink	(300,300,100)	340-260=80	340-260=80	200-0=200

The algorithm for this example is run three times to attempt to lower the cost of the trajectories even more than what can be done with one run of the algorithm. When the algorithm is run and the random dense tree (RDT) is grown, all the paths generated must utilize this tree in their materialization. Since the tree is grown randomly in the space, there is a chance that running the algorithm again will produce a RDT that will provide even lower cost trajectories.

The algorithm generates an initial sub-optimal solution as a combination of tree-expanded path and direct-solution path. After the iteration count reaches fruition, a total of 9 paths are created, each with a decreased cost as measured by total time to traverse from the initial configuration to the goal configuration. The cost for the final solution of the first run is 21.9s. The vehicle approaches the region but then turns to the left to avoid the large central obstacle, climbs, and turns to the right where a clear path to the goal location and heading can be made using the 3-D Dubins airplane. The final path is produced after 15.8s of computing time and is shown along with the complete RDT associated with it in Figure 5-2.

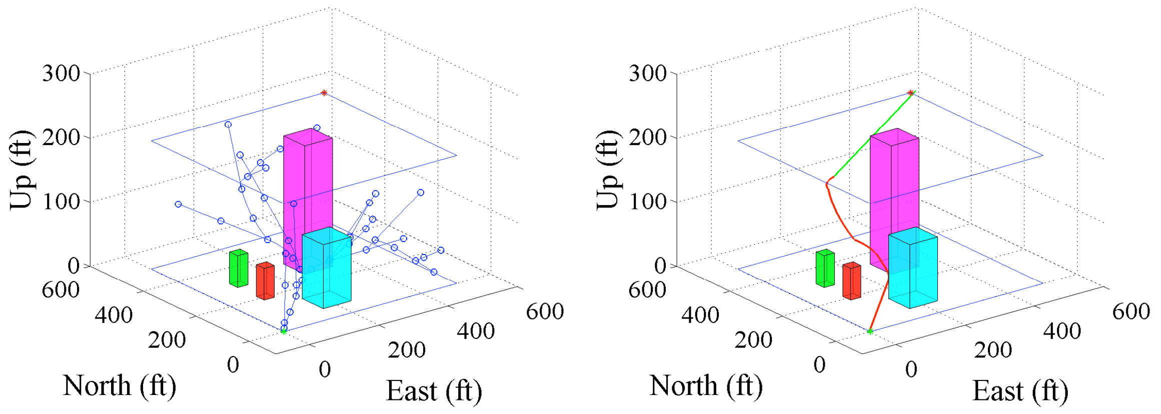


Figure 5-2. Fully grown RDT (left) and final sub-optimal path (right) of run #1 of motion planning algorithm in urban environment with time-traveled cost of 21.9s.

The simulation is run again with a new RDT. After all of the iterations are complete, 10 more sub-optimal solutions are generated, each with a smaller total travel time than the one prior. The tenth path is determined after 16.7s of computation. The final solution results in a path shown in Figure 5-3 that includes a cost of 21.2s. This path also turns to the left to avoid the center obstacle but climbs at a higher rate, then utilizes the Dubins airplane to reach the goal configuration.

A third and final run of the simulation, with another new random dense tree, generates only 5 sub-optimal solutions with decreasing total cost, the final being found after only 9.6s of computation time. In this case, the final solution requires only 18.8s

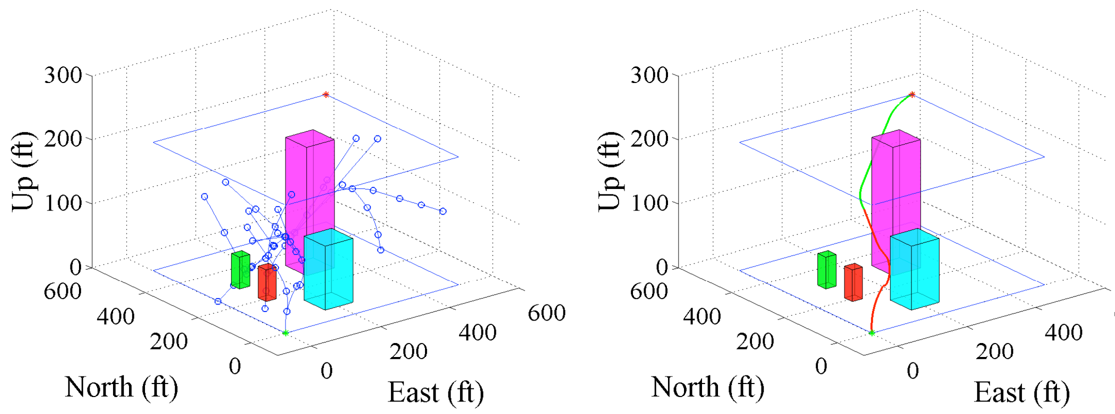


Figure 5-3. Fully grown RDT (left) and final sub-optimal path (right) of run #2 of motion planning algorithm in urban environment with time-traveled cost of 21.2s.

to follow the path shown in Figure 5-4. This solution performs an almost straight path toward the goal point, turning slightly to the left to avoid the central obstacle, all while implementing a larger climb rate. Once past the obstacle, the optimal portion of the solution is implemented.

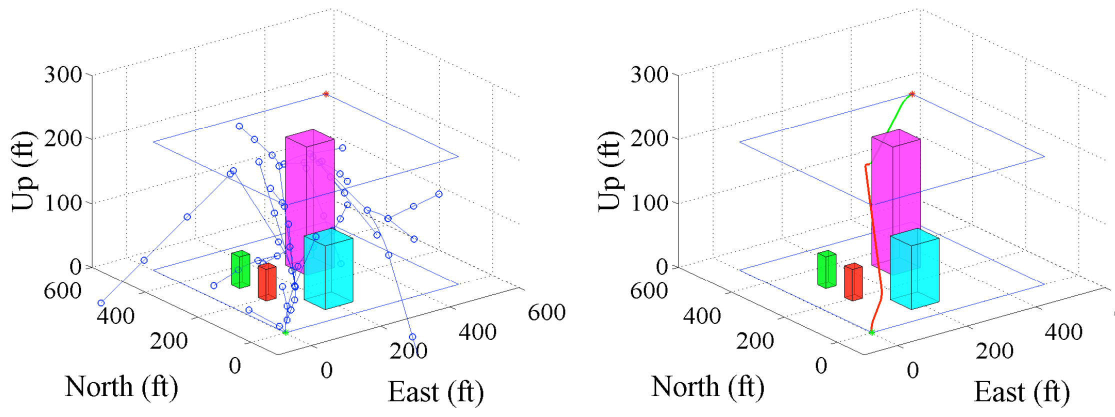


Figure 5-4. Fully grown RDT (left) and final sub-optimal path (right) of run #3 of motion planning algorithm in urban environment with time-traveled cost of 18.8s.

In the case of running these three simulations, the third run of the simulation provides the solution with the lowest cost of 18.8s of travel time. It appears that this is due to the utilization of the larger climb rate and the fact that the path is almost straight between the initial and goal points. It should be noted that the higher climb rate implemented in this final run is still lower than the vehicles maximum climb rate.

5.4 Random Dense Tree Algorithm Modification

In this example, the full motion planning algorithm is utilized for the same circumstances in Section 5.3 (initial configuration, desired goal configuration, obstacle locations, and obstacle dimensions), but some of the attributes of the RDT development are altered. This algorithm runs for twice as many iterations allowing for the RDT to grow larger and with more branches. In addition, the randomized node placement is prescribed to be influenced by the goal point to a greater degree. The algorithm for this example is also run three times with the purpose of potentially producing a RDT that will yield lower cost trajectories. The solutions to all the simulations are presented as follows.

Upon completion of the first run of the algorithm, a total of 4 paths are created, each with a decreased cost. The final path is determined after 25.1s of computing time. The cost for the final solution of the first run is 19.6s. The vehicle turns left immediately and climbs along the western boundary of the $\mathcal{C}$-space and then turns right between the two smaller obstacles where the optimal portion of the algorithm is implemented to create the second portion of the path. This final path and the complete RDT associated with it are shown in Figure 5-5.

The simulation is run a second time with a new RDT and after all of the iterations are completed, 11 more sub-optimal solutions are generated, each with a smaller total travel time than the one prior. The eleventh and final path is determined after 24.0s of computation and it is shown in Figure 5-6 with a travel time of 18.6s. This path resembles the final path of the third run in Section 5.3 as it performs an almost straight path toward the goal point, turning slightly to the left to avoid the central obstacle, all while implementing an even larger climb rate. Once past the obstacle, the optimal portion of the solution is implemented.

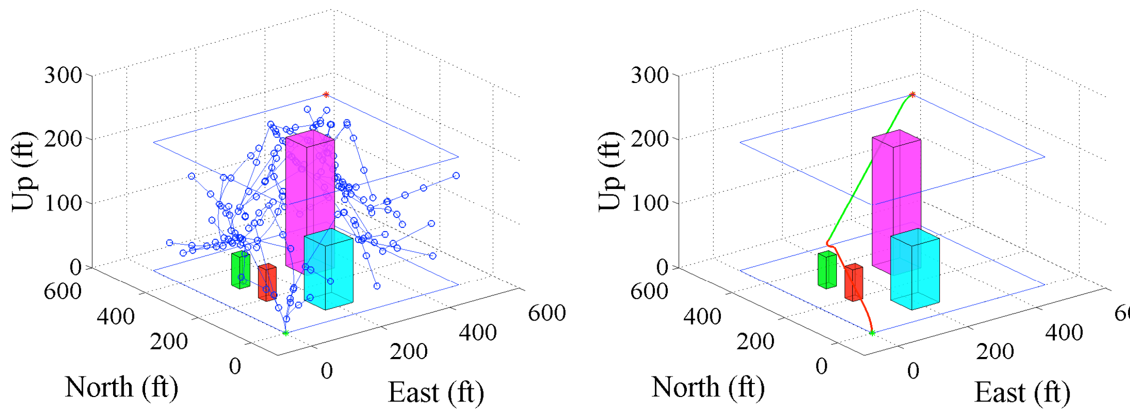


Figure 5-5. Fully grown RDT (left) and final sub-optimal path (right) of run #1 of motion planning algorithm with RDT formulation adjustment in urban environment with time-traveled cost of 19.6s.

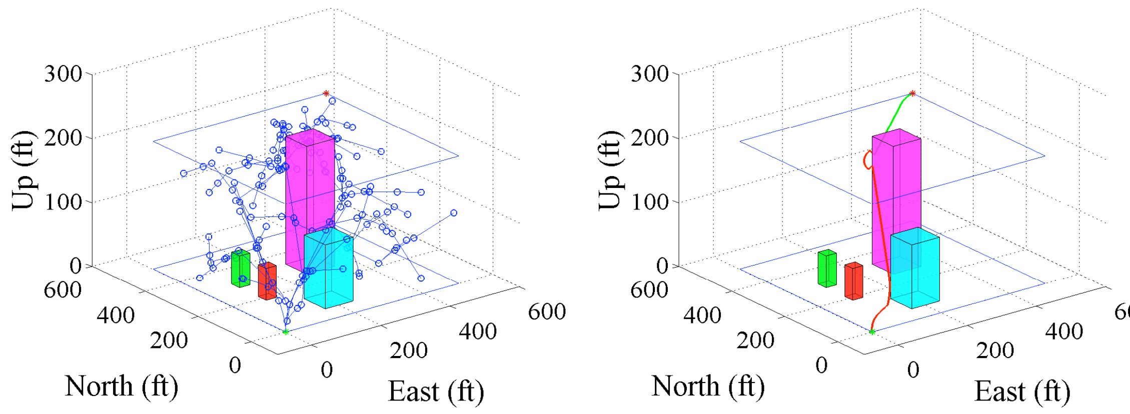
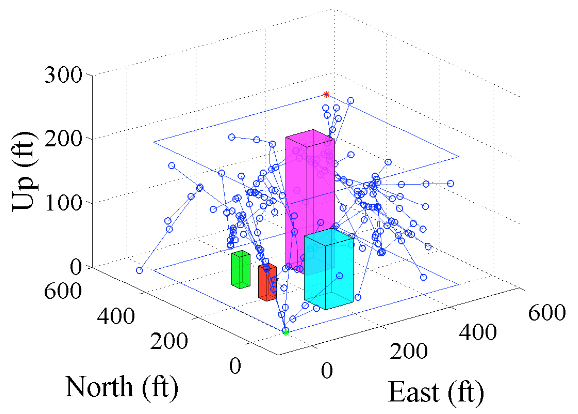


Figure 5-6. Fully grown RDT (left) and final sub-optimal path (right) of run #2 of motion planning algorithm with RDT formulation adjustment in urban environment with time-traveled cost of 18.6s.

The third and final run of the simulation, with another new RDT, also generates 11 sub-optimal solutions, each with decreasing total cost. 47.1s of computation time is needed for this run of the algorithm to find the final solution. A cost of 19.0s is required to follow the final path shown in Figure 5-7. The path generated in this solution starts climbing to the north-northeast and travels over the first small obstacle, continues climbing while turning to the east and then switches to the optimal portion of the solution.

In the case of running these three simulations, the second run of the simulation provides the solution with the lowest cost of 18.6s of travel time. Similar to the final run



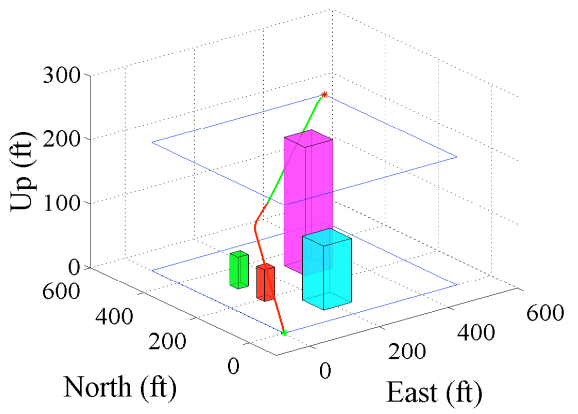


Figure 5-7. Fully grown RDT (left) and final sub-optimal path (right) of run #3 of motion planning algorithm with RDT formulation adjustment in urban environment with time-traveled cost of 19.0s.

of the algorithm in Section 5.3, the even larger climb rate and general heading of the path provides the favorable performance.

In comparison, there are some similarities and differences in the performance of these two versions of the algorithm. These are presented in Table 5-3. The altered algorithm attains the shortest overall path travel time of 18.6s and a shorter average path travel time of 19.07s. The average computing time more than doubles when using the altered algorithm verses the original algorithm while the performance only improves slightly.

Table 5-3. Motion planning algorithm results comparison.

Algorithm	Shortest Overall Travel Time	Average Overall Travel Time	Average Computation Time
Original	18.8s	20.63s	14.03s
Altered	18.6s	19.07s	32.07s

5.5 Full Motion Planning in Urban Environment With Bridges

In this example, the original full motion planning algorithm is utilized for an urban environment that includes a covered walkway and an elevated bridge. The vehicle starts at a position of (0,0,0) and a heading of 60^o while it is required to end at a position of (500,500,200) and a heading of 90^o. The environment consists of 2 large towers, 1 small

Table 5-4. Tower, walkway, and bridge dimensions and locations. All units in feet.

Obstacle	Coordinates of Center	dx	dy	dz
North Tower	(50,250,100)	100-0=100	300-200=100	200-0=200
Covered Walkway	(175,275,25)	250-100=150	300-250=50	50-0=50
Northeast Tower	(275,275,100)	300-250=50	300-250=50	200-0=200
Elevated Bridge	(275,175,175)	300-250=50	250-100=150	200-150=50
East Tower	(250,50,100)	300-200=100	100-0=100	200-0=200

tower, a covered walkway, and an elevated bridge. These obstacles must be strictly avoided just as in the other examples. The details of the obstacles are presented in Table 5-4.

Just as in the examples in Sections 5.3 and 5.4, the tree portion of the motion planning algorithm is utilized for maneuvering around the obstacles and the 3-dimensional Dubins airplane sequence is utilized for traveling from the end of the branches to the goal point and heading. The difference with this case is that the path is encouraged to fly over the covered walkway and/or under the elevated bridge. This demonstrates that the motion planner treats the obstacles as 3-dimensional objects rather than 2-dimensional regions.

The algorithm for this example is only run once as the intention is to demonstrate the motion planning capabilities of the algorithm, not the performance as in Sections 5.3 and 5.4. Two of the paths created in this simulation, along with the final RDT, are presented in Figure 5-8. Notice that the trajectory paths generated indeed have the ability to fly over obstacles, as in the case of the covered walkway, and under obstacles, as in the case of the elevated bridge.

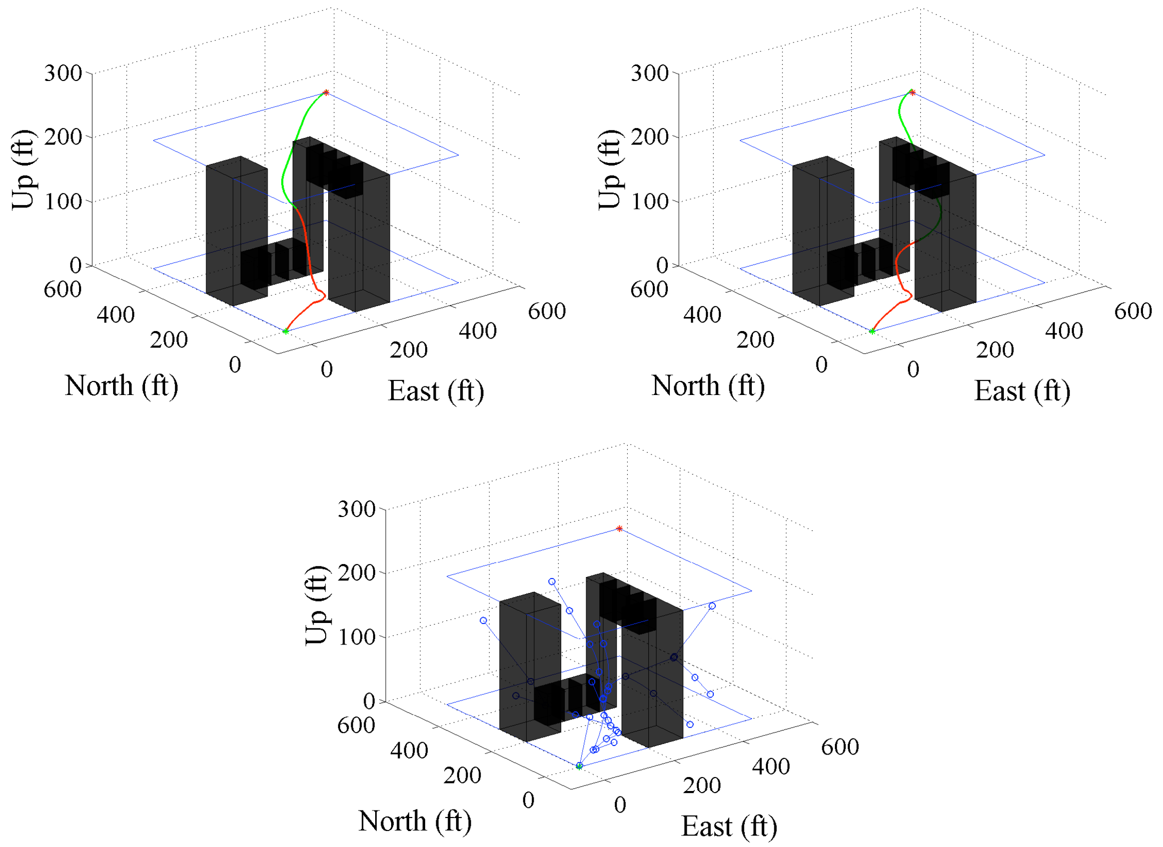


Figure 5-8. The motion planning algorithm treats the obstacles as 3-dimensional objects rather than 2-dimensional regions. Generated paths can go over the covered walkway (top left) and under the elevated bridge (top right). The final RDT (bottom) is also shown growing over, under, and around the obstacles.

CHAPTER 6
CONCLUSION

The work presented in this thesis has addressed the problem of motion planning in 3-dimensional space. Specifically, motion planning for missions involving vehicles with 3-dimensional motion in close proximity to 3-dimensional obstacles was developed. The topic of 3-dimensional motion primitives was addressed in Chapter 2 with a development of a 3-dimensional version of the Dubins car called the Dubins airplane. This Dubins airplane implemented a constant climb rate in the motion primitive definitions to expand into the third dimension. In addition, the 2-primitive turn-straight path segments were discussed and developed for 3-dimension space. Chapter 3 focuses on the subject of random dense trees (RDTs). The theory of randomized methods for path planning was touched upon briefly, followed by the concept of 2-dimensional RDTs presented by Kehoe [18]. The general idea of how RDTs grow was presented and then both rapidly-exploring random tree theory and expansive-spaces tree theory were described and discussed. Following the introduction of 2-dimensional RDT theory, 3-dimensional RDT theory was introduced utilizing the rapidly-exploring random tree theory but altered for 3-dimensional growth. Finally the topic of 3-dimensional RDT trajectory generation was discussed in full.

In Chapter 4, the ideas of 3-dimensional motion primitives, both the 2-primitive turn-straight combinations and 3-primitive Dubins airplane, and the 3-dimensional RDT concepts were brought together into the motion planning algorithm that was developed for this thesis. An explanation of how the algorithm functions was presented and discussed in detail. This motion planning algorithm was then put through 4 examples in Chapter 5. The first example was for an obstacle-free environment and only the second portion of the motion planning algorithm was implemented. In the second example, the full motion planning algorithm was put to use for a vehicle traveling through an obstacle-rich environment. The third example was conducted to exhibit the difference in performance of the original RDT growth algorithm and an altered RDT growth algorithm that grew for twice as long and was more influenced by the goal configuration. Lastly, an example was presented that showcased the motion planning algorithm's ability to not only plan trajectories around obstacles, but over and under them as well in the case of a series of bridges.

There are several directions that can be considered for future work using this research. Implementation of the path parameterization work completed by Pachikara [25] into the motion planning algorithm presented in this thesis is being employed for progression of this research. The extension of this investigation for use in a sensor-based motion planning scheme is a logical one and will also be investigated. The cooperation of the field of flight controls with this motion planning algorithm is also a potential area of interest for future work.

REFERENCES

[1] Choset, H., Lynch, K., Hutchinson, S., Kantor, G., Burgard, W., Kavraki, L., and Thrun, S., *Principles of Robot Motion: Theory, Algorithms, and Implementations*, MIT Press, Cambridge, MA, 2005.

[2] Faiz, N., Agrawal, S., and Murray, R., "Trajectory Planning of Differentially Flat Systems with Dynamics and Inequalities," *Journal of Guidance, Control, and Dynamics*, Vol. 24, No. 2, 2001, pp. 219-227.

[3] Fliess, M., Lévine, J., Martin, P., and Rouchon, P. "Flatness and Defect of Nonlinear Systems: Introductory Theory and Examples," *International Journal of Control*, Vol. 61, No. 6, 1995, pp.1327-1361.

[4] Van Nieuwsadt, M.J., and Murray, R.M., "Real-Time Trajectory Generationfor Differenially Flat Systems," *International Journal of Robust and Nonlinear Control*, Vol. 8, No. 11, 1998, pp. 1995-1020.

[5] Kuwata, Y., and How, J., "Three Dimensional Receding Horizon Control for UAVs," *Proceedings of the 2004 AIAA Guidance, Navigation, and Control Conference*, Providence, RI, August 2004.

[6] Richards, A., and How, J., "Aircraft Trajectory Planning with Collision Avoidance Using Mixed Integer Linear Programming," *Proceedings of the 2002 IEEE American Control Conference*, Anchorage, AK, May 2002.

[7] Schouwenaars, T., How, J., and Feron, E., "Receding Horizon Path Planning with Implicit Safety Guarantees," *Proceedings of the 2004 IEEE American Control Conference*, Boston, MA, June 2004.

[8] Frazzoli, E., Dahleh, M., and Feron, E. "Maneuver-Based Motion Planning for Nonlinear Systems with Symmetries," *IEEE Transactions on Robotics*, Vol. 21, No. 6, December 2005.

[9] Frazzoli, E., Dahleh, M., and Feron, E., "Real-Time Motion Planning for Agile Autonomous Vehicles," *Journal of Guidance, Control, and Dynamics*, Vol. 25, No. 1, January-February 2002.

[10] Hsu, D., Latombe, J.C., and Motwani, R., "Path Planning in Expansive Configuration Spaces," *Proceedings of the IEEE Conference on Robotics and Automation*, 1997.

[11] 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 and Automation*, Vol. 12, No. 4, August 1996, pp. 566-580.

[12] LaValle, S.M., and Kuffner, J.J., "Randomized Kinodynamic Planning," *International Journal of Robotics Research*, Vol. 20, No. 5, May 2001, pp. 378-400.

[13] Dubins, L., "On Curves of Minimal Length with a Constraint on Average Curvature and with Prescribed Initial and Terminal Positions and Tangents," *American Journal of Mathematics*, Vol. 79, No. 1, 1957, pp. 497-516.

[14] Shanmugavel, M., Tsourdos, A., Zbikowski, R., and White, B. A., "3D Dubins Sets Based Coordinated Path Planning for Swarm of UAVs"*AIAA Guidance, Navigation, and Control Conference*, Keystone, Colorado, 2006.

[15] Chitsaz, H., and LaValle, S. M., "On Time-optimal Paths for the Dubins Airplane" *2007 IEEE Conference on Decision and Control* New Orleans, LA, 2007.

[16] Hwangbo, M., Kuffner, J. and Kanade, T., "Efficient Two-Phase 3D Motion Planning for Small Fixed-Wing UAVs," *IEEE International Conference on Robotics and Automation*, April 2007, WeC12.1.

[17] Sasiadek, J., and Duleba, I., "3D Local Trajectory Planner for UAV," *Journal of Intelligent and Robotic Systems*, Vol. 29, 2000, 191-210.

[18] Kehoe, J., "Trajectory Generation for Effective Sensing in Close Proximity Environments," *Ph.D. Dissertation*, University of Florida, August 2007.

[19] Howlett, J., Goodrich, M., and McLain, T., "Learning Real-Time A^* Path Planner for Sensing Closely-Spaced Targets from an Aircraft," *Proceedings of the 2003 AIAA Guidance, Navigation, and Control Conference*, Austin, TX, August 2003.

[20] Le Ny, J., and Feron, E., "An Approximation Algorithm for the Curvature-Constrained Traveling Salesman Problem," *Proceedings of the* 43^{rd} *Annual Allerton Conference on Communications, Control, and Computing*, September 2004.

[21] McGee, T.G. and Hedrick, J.K., "Optimal Path Planning with a Kinematic Airplane Model," *Journal of Guidance, Control, and Dynamics*, Vol. 30, No. 2, March-April 2007.

[22] Tang,Z. and Ozguner, U., "Motion Planning for Multitarget Surveillance with Mobile Sensor Agents," *IEEE Transactions on Robotics*, Vol. 21, No. 5, October 2005, pp. 898-908.

[23] Yang, G., and Kapila, V., "Optimal Path Planning for Unmanned Vehicles with Kinematic and Tactical Constraints," *Proceedings of the* 41^{st} *IEEE Conference on Decision and Control*, Las Vegas, NV, December 2002.

[24] Shkel, A., and Lumelsky, V., "Classification of the Dubins Set," *Robotics and Autonomous Systems*, Vol. 34, 2001, pp. 179-202.

[25] Pachikara, A., Kehoe, J., Lind, R., "A Path-Parametrization Approach using Trajectory Primitives for 3-Dimensional Motion Planning," *AIAA Guidance, Navigation and Control Conference*, August 2009.

[26] Ferguson, D., Nidhi, K., and Stentz, A., "Replanning with RRTs," *Proceedings of the IEEE International Conference on Robotics and Automation*, May 2006, pp. 1243-1248.

[27] Kalisiak, M., and Van de Panne, M., "RRT-Blossom: RRT with a Local Flood-Fill Behavior," *Proceedings of the IEEE International Conference on Robotics and Automation*, May 2006, pp. 1237-1242.

[28] Kuffner, J., and LaValle, S., "RRT Connect: An efficient Approach to Single-Query Path Planning," *Proceedings of the IEEE International Conference on Robotics and Automation*, 2000, pp. 995-1001.

[29] Melchior, N., and Simmons, R., "Particle RRT for Path Planning with Uncertainty," *Proceedings of the IEEE International Conference on Robotics and Automation*, April 2007, pp. 1617-1624.

[30] Phillips, J., Bedrossian, N., and Kavraki, L., "Guided Expansive Spaces Trees: A Search Strategy for Motion- and Cost-Constrained State-Spaces," *Proceedings of the 2004 IEEE International Conference on Robotics and Automation*, New Orleans, LA, April 2004.

[31] Strandberg, M., "Augmenting RRT-Planners with Local Trees," *Proceedings of the IEEE International Conference on Robotics and Automation*, Vol. 4., April 2004, pp. 3258-3262.

[32] Hsu, D., Kindel, R., Latombe, J.C., and Rock, S., "Randomized Kinodynamic Motion Planning with Moving Obstacles," *International Journal of Robotics Research*, Vol. 21, No. 3, 2002, pp. 233-255.

[33] Laumond, J.P., *Robot Motion Planning and Control*, Online book: <http://www.laas.fr/ jpl/book.html>.

BIOGRAPHICAL SKETCH

Ryan Donovan Hurley was born in 1982 in Fort Myers, FL. He grew up in Fort Myers Beach, FL and graduated from Cypress Lake High School Center for the Arts in 2001. He then attended the University of Florida where he received Bachelor of Science degrees in both aerospace engineering and mechanical engineering in December of 2006. The following month, he began work for Attractions and Engineering Services at Walt Disney World. He left the company to return to the Department of Mechanical and Aerospace Engineering at the University of Florida in January of 2008 to pursue graduate school. His research in the Flight Control Laboratory has focused on the maturation of autonomy-enabling guidance and control technologies for small unmanned aerial vehicles (UAVs). He received his Master of Science degree from the University of Florida in the spring of 2009 in aerospace engineering. After pursuing his doctoral degree, he plans to work toward a successful career in controls engineering in the field of aerospace technology.