The Pennsylvania State University
The Graduate School

ENERGY AWARE PATH PLANNING IN COMPLEX FOUR DIMENSIONAL ENVIRONMENTS

A Dissertation in
Aerospace Engineering
by
Anjan Chakrabarty

Submitted in Partial Fulfillment
of the Requirements
for the Degree of

Doctor of Philosophy

August 2014

The dissertation of Anjan Chakrabarty was reviewed and approved* by the following:

Jacob W Langelaan
Associate Professor of Aerospace Engineering
Dissertation Advisor and Chair of Committee

Mark D. Maughmer
Professor of Aerospace Engineering

Joseph F. Horn
Associate Professor of Aerospace Engineering

Constantino Lagoa
Professor of Electrical Engineering

George A. Lesieutre
Professor of Aerospace Engineering
Head of Aerospace Engineering

*Signatures are on file in the Graduate School.

Abstract

This dissertation addresses the problem of energy-aware path planning for small autonomous vehicles. While small autonomous vehicles can perform missions that are too risky (or infeasible) for larger vehicles, the missions are limited by the amount of energy that can be carried on board the vehicle. Path planning techniques that either minimize energy consumption or exploit energy available in the environment can thus increase range and endurance. Path planning is complicated by significant spatial (and potentially temporal) variations in the environment. While the main focus is on autonomous aircraft, this research also addresses autonomous ground vehicles.

Range and endurance of small unmanned aerial vehicles (UAVs) can be greatly improved by utilizing energy from the atmosphere. Wind can be exploited to minimize energy consumption of a small UAV. But wind, like any other atmospheric component , is a space and time varying phenomenon. To effectively use wind for long range missions, both exploration and exploitation of wind is critical.

This research presents a kinematics based tree algorithm which efficiently handles the four dimensional (three spatial and time) path planning problem. The *Kinematic Tree* algorithm provides a sequence of waypoints, airspeeds, heading and bank angle commands for each segment of the path. The planner is shown to be resolution complete and computationally efficient. Global optimality of the cost function cannot be claimed, as energy is gained from the atmosphere, making the cost function *inadmissible*. However the Kinematic Tree is shown to be optimal up to resolution if the cost function is admissible. Simulation results show the efficacy of this planning method for a glider in complex real wind data. Simulation results verify that the planner is able to extract energy from the atmosphere enabling long range missions.

The Kinematic Tree planning framework, developed to minimize energy consumption of UAVs, is applied for path planning in ground robots. In traditional path planning problem the focus is on obstacle avoidance and navigation. The optimal Kinematic Tree algorithm named Kinematic Tree* is shown to find optimal paths to reach the destination while avoiding obstacles. A more challenging path planning scenario arises for planning in complex terrain.

This research shows how the Kinematic Tree* algorithm can be extended to find minimum energy paths for a ground vehicle in difficult mountainous terrain.

Table of Contents

List of Figures ix

List of Tables xii

List of Symbols xiii

Acknowledgments xvi

Chapter 1
Introduction 1
1.1 Motivation . . . 3
1.2 Related Work and Previous Work . . . 6
1.2.1 Flight of Birds . . . 7
1.2.2 Path Planning in a Flow field . . . 8
1.2.3 Autonomous Soaring . . . 9
1.2.4 Graph-based Approaches to Flight Path Planning . . . 10
1.2.4.1 Previous Work . . . 11
1.2.5 Sampling Based Planning . . . 13
1.2.5.1 Sampling based planners with cost . . . 14
1.3 Overview of Research . . . 15
1.4 Contributions . . . 16
1.5 Readers Guide . . . 17

Chapter 2
Sampling Based Motion Planning for Energy Aware Flight 18
2.1 Problem Formulation . . . 19
2.1.1 Planning with Cost . . . 21
2.1.2 Planning with Power Ups . . . 21

2.2 General Framework for Sampling based Methods . . . 22
2.3 Existing Algorithms . . . 23
2.3.1 Probabilistic Roadmaps (PRM) . . . 23
2.3.2 Rapidly-exploring Random Trees (RRT) . . . 25
2.3.2.1 Resolution Complete RRT (RC-RRT) . . . 25
2.3.2.2 RRT* . . . 26
2.3.3 Time-Varying problems . . . 28
2.4 Summary: The Planning Problem . . . 29

Chapter 3
The Kinematic Tree **30**
3.1 Kinematic Tree Algorithm . . . 31
3.1.1 Motion Primitives . . . 33
3.1.2 Resolution Completeness . . . 34
3.1.3 Time Complexity . . . 36
3.1.3.1 Time Complexity of KT . . . 36
3.1.3.2 Time Complexity Analysis of RRT . . . 36
3.2 Optimal Path planning . . . 38
3.2.1 Admissibility of Cost Function . . . 39
3.2.2 Optimality of Kinematic Tree* Algorithm . . . 40
3.3 Heuristic Function and Search Efficiency . . . 42
3.3.1 Adjusting weights of g and h . . . 43
3.3.2 Probabilistic Analysis of Complexity of KT* . . . 44
3.3.2.1 A General Formula for Mean Complexity of KT* . . . 45
3.3.2.2 Bounds on Complexity . . . 47
3.4 Summary . . . 48

Chapter 4
Kinematic Tree For Flight Planning **49**
4.1 Mountain Wave . . . 50
4.1.1 Meteorological Modeling . . . 51
4.2 Kinematics of Soaring Flight . . . 53
4.3 The Kinematic Tree Construction for Path Planning . . . 56
4.3.1 Motion Primitives . . . 56
4.3.2 A note on Resolution . . . 60
4.3.3 Node Selection and Expansion . . . 60
4.3.4 End Game . . . 62
4.4 Simulation Results . . . 63
4.4.1 Feasible Paths to the Goal using Wind energy . . . 63
4.4.2 Maximum Altitude Gain by Nimbus III DM . . . 69

4.5 Summary . . . 71

Chapter 5
Kinematic Tree for Optimal Path Planning **75**
5.1 Obstacle Avoidance . . . 76
5.1.1 The Algorithm . . . 76
5.1.1.1 The Unicycle Car Model . . . 77
5.1.2 Varying the weights of g and h . . . 79
5.1.3 Dynamic Weighting . . . 83
5.2 Path Planning on Mountainous Terrain . . . 84
5.2.1 Vehicle kinematics and motion primitives . . . 84
5.2.2 Node selection and endgame region . . . 86
5.2.3 Simulation results . . . 87
5.3 Summary . . . 93

Chapter 6
Conclusion **94**
6.1 Summary of Contributions . . . 95
6.1.1 Kinematic Tree Algorithm . . . 95
6.1.2 Kinematic Tree* Algorithm . . . 96
6.1.3 Environment aware planning . . . 96
6.2 Future Works . . . 97
6.2.1 Unknown wind field . . . 97
6.2.2 Applications . . . 98
6.2.2.1 Science Missions . . . 98
6.2.2.2 Next Generation Air Transportation . . . 99
6.2.2.3 Autonomous Rovers . . . 99

Appendix A
Green Flight Challenge **100**
A.1 Performance Data of Taurus G4 . . . 100
A.2 Graph Based Planning and the Green Flight Challenge . . . 102
A.3 Graph Based Planning Technique . . . 103
A.4 The Competition . . . 103
A.5 Flight Between Nodes . . . 104
A.6 Flight Planning . . . 107
A.6.1 Segment by segment optimization . . . 107
A.6.2 Trajectory refinement . . . 110
A.6.3 The utility of flight planning . . . 113
A.7 Competition Flights . . . 113

A.7.1 Efficiency flight: September 27, 2011 114
A.7.1.1 Atmospheric conditions 114
A.7.1.2 Planned and actual flight conditions 114
A.7.2 Speed flight: September 29, 2011 114
A.7.2.1 Atmospheric conditions 114
A.7.2.2 Planned and actual flight conditions 115
A.7.3 Competition results 115
A.8 Pros and Cons of Graph Based Planning 117
A.9 Summary 117

Appendix B
Vehicle Properties of SB-XC **120**
B.1 The Nimbus III DM 121
B.1.1 Calculation of drag polar for Nimbus 121

Bibliography **124**

List of Figures

1.1 Schematic of Mission Scenario: The aircraft is looking to find minimum energy routes through a given wind field to reach a destination. 2
1.2 Ground vehicle navigating through difficult terrain. Image source: $https://www.peruadventurestours.com$ 3
1.3 Practical applications where wind energy is used for long distance travel. . . . 4
1.4 Different types of updrafts used for static soaring. 5
1.5 Bird Tracks of North America. Turkey vulture in orange; golden eagle in blue. [1]. 7
1.6 The Zermelo Problem. 8
1.7 Schematic of graph-based planning for autonomous soaring over a wind field. 11
1.8 Flight paths for energy map and A*. Left side: paths at 1000m altitude; right side: paths at 2000m altitude [2]. 12
1.9 Probabilistic Road Map and Rapidly Exploring Random Tree. Image Source: [3]. 13

2.1 Schematic of Mission Scenario. 19
2.2 Probabilistic Roadmap: Growth of the connected components in the free space 23
2.3 Comparison between RRT and RRT* [4]. 26

3.1 Growth of Kinematic Tree 31
3.2 Comparison of growth of RC- RRT (upper) and Kinematic Tree (lower) . . . 34
3.3 Consistency Condition. 40
3.4 Situation when (n+1)th node is selected. 41
3.5 Number of nodes expanded vs branches. 43
3.6 Tree model for analyzing complexity of KT*. The goal node is located at a depth of N in the tree. The branches in red denote "of course" branches that does not lead to the solution. 45

4.1 A wave over the Bald Eagle Valley of central Pennsylvania. Source: Wikipedia 50
4.2 Visualization of wind field data for a complex time varying wind field. 52
4.3 Point mass model. 54

4.4 Motion Primitives in zero Wind. 58
4.5 Spiral Motion Primitives in zero Wind. 59
4.6 Endgame Region. 62
4.7 Scenario For simulation. The goal is shown by the black dot and start regions are shown by blue,yellow and red dots. 63
4.8 Trajectories starting at Different Times of Day. 64
4.9 Flight starting at 1100 UTC for *yellow* starting position. 67
4.10 Flight starting at 1100 UTC for *yellow* starting position: time history of velocity, heading, vertical component of wind and altitude. 68
4.11 The Nimbus III-DT. Source: http : //francoise.fischer.free.fr/n3/d_kexx.jpg . 69
4.12 Altitude Gain by Nimbus III-DM using only wave 71
4.13 Time Evolution of Maximum Altitude Gain Flight. 72
4.14 Altitude Gain by Nimbus III-DM using only wave: time history of velocity, heading, vertical component of wind and altitude. 73

5.1 Path discovered by Kinematic Tree* Algorithm. 79
5.2 Path discovered by Kinematic Tree* Algorithm with $\alpha = 1$. 80
5.3 Comparison between the paths found by varying α.Upper Left: Path found by$\alpha = 1$; Upper Right: Path found by $\alpha = 0.8$. Down Left: Paths found by $\alpha = 0.6$. Down right: Paths found by $\alpha = 0.5$ 82
5.4 Path discovered by dynamic weighted Kinematic Tree Algorithm. 83
5.5 Motion Primitives calculated at zero elevation are projected onto the terrain. . 86
5.6 Efficiency as a function of percentage of load. 88
5.7 Minimum energy Path for Ground vehicle over Central Pennsylvania. 89
5.8 Velocity, heading and power consumed for the red path. 90
5.9 Velocity, heading and power consumed for the green path. 91
5.10 Velocity, heading and power consumed for the blue path. 92

A.1 External dimensions and configuration of the Taurus G4 101
A.2 Aircraft performance data and curve fits for drag polar (left) and propeller (right) . Data is shown as points, the fitted curve curve is shown as solid line. . 102
A.3 Green Flight Challenge course looking west. Paddles show GPS waypoints; brake release occurs at the paddle labelled "S;" the finish line is at the paddle labelled "G." 104
A.4 Track coordinate frames (left) and resolution of airspeed and wind vectors into the track coordinate frame (right). Positive angles are shown. 105
A.5 Node placement and in track winds in the course. 108
A.6 Turnpoints (squares) and nodes (circles) used for flight planning. 109

A.7 Power Setting, Indicated Airspeed, and Ground Speed for the energy flight. The grey line shows the flight condition after the segment by segment optimization, the black line line shows flight condition after trajectory refinement. 111
A.8 Predicted minimum required energy consumption for the energy flight. The grey line shows energy consumption after the segment by segment optimization, the black line line shows energy consumption after trajectory refinement. 112
A.9 Comparison of energy-optimized flight plan with constant ground speed flight plan. 113
A.10 Flight data and trace of energy flight. Left: Ground speed, power and energy consumption for efficiency flight. Solid black line shows data from eTotalizer, grey line is planned flight condition, open circles show pilot observed energy consumption; Right: trace of GPS positions. 115
A.11 Flight data and trace of speed flight. Left: Ground speed, power and energy consumption for speed flight. Solid black line shows data from eTotalizer, grey line is planned flight condition, open circles show pilot observed energy consumption; Right: trace of GPS positions. 116
A.12 Predicted winds over the course at 4000 feet MSL for September 27, 2011 (efficiency flight). 118
A.13 Winds over the course at 4000 feet MSL for September 29, 2011 (speed flight). 119

B.1 SB-XC Glider. Image source: http://www.rcgroups.com/forums/. 120
B.2 Sink rate vs. airspeed for the SB-XC. Minimum sink is approximately 0.56 m/s and occurs at approximately 14.6 m/s. 121
B.3 The Nimbus III-DT glider. 122
B.4 The Nimbus III-DM polar plots. 122

List of Tables

4.1 Parameters for SB-XC glider. 64
4.2 Simulation Results for Different Start Points. 65
4.3 Time of travel and Minimum Distance to goal for trajectories starting at different times of the day for the *yellow* start point 65
4.4 Time of travel and Minimum Distance to goal for trajectories starting at different times of the day for the *yellow* start point 66
4.5 Parameters for Nimbus III DM. 70

5.1 Comparison of KT* algorithms for different values of α 81
5.2 Parameters for Ground Vehicle 87
5.3 Ground vehicle path costs, distances traveled and time taken to reach the destination for both energy-optimal path and straight line path. 89

A.1 Taurus G4 basic data[5] 101
A.2 Turnpoint locations and sequence 104
A.3 Final results of 2011 Green Flight Challenge[6]. Energy consumption and speeds are from CAFE Foundation measuring equipment. 116

B.1 Parameters for SB-XC glider. 121
B.2 Parameters for Nimbus III DM. 121
B.3 Speed Polar for Nimbus III DM 123

List of Symbols

α_i	incidence angle between thrust vector and flight path
γ	glide path angle
η	rate of change of heading
η_{ec}	efficiency of energy conversion
η_p	propeller efficiency
ρ	density
σ	path
ϕ	bank angle
ψ	heading angle
C_D	Drag Coefficient
C_L	Lift Coefficient
C_{rr}	rolling friction coefficient.
c_d	aerodynamic drag constant for ground vehicle
D	Drag
E	Edge Set
e	Specific energy
F_r	Frictional force

G	Graph $G = (V, E)$ on X is composed of vertex set V and an edge set E
g	acceleration due to gravity
h	altitude
J	advance ratio
L	Lift
L/D	ratio of Lift to Drag
m	mass
n	propeller speed in revolutions per second
$O(.)$	The function $f(n)$ is said to be $O(g(n))$, denoted as $f(n) \in O(g(n))$, if there exists two constants M and n_0 such that $f(n) \leq Mg(n)$ for all $n \geq n_0$.
P	Power
p	wing span
Re	Reynolds number
S	wing area
T	Thrust
t	time
V	Vertex Set
v_a	air speed
v_g	ground speed
w	wind
W	Work done
X	Configuration Space also State Space
X_{free}	Free region such that $X_{free} = X/X_{obs}$
X_{goal}	Goal region such that $x_{goal} \in X_{goal}$

X_{obs}	Obstacle region
x_{goal}	final configuration
x_{init}	initial configuration

Acknowledgments

I sincerely believe life is all about the choices you make. Some very intelligent person have said: “when a person really desires something, all the universe conspires to help him realize his dream”. The problem lies in: do you really know what you want? For me the decision to pursue a career in aerospace engineering was more about working in certain areas than to focus on getting a higher degree. The process of getting a doctorate degree came along realizing that dream. But as it turned out: “nobody said it was easy” but,“no one ever said it would be this hard”. A good support system is very important to survive grad school.

The single most important factor in determining your research and quality of life as a grad student depends on your advisor. I had the very best one can hope for: Dr. Jack Langelaan. Jack has a solution to all the problems under the sun: from research to proposals, from jobs to visa issues. Jack constantly motivates you to find innovative solutions to problems. The constant guidance, albeit the intellectual freedom, brings the best out of everyone in our research group. If I pursue a career in academia I would like to emulate some of his characteristics as an advisor myself.

I would sincerely like to thank each member of my dissertation committee: Dr. Constantino Lagoa for providing very useful thoughts on theoretical insights in the research; Dr. Joseph Horn was quick to point out the additional constraints that need to be considered for successful implementation of the algorithms in real life; Dr. Mark D. Maughmer, having all the experience of flying over Central Pennsylvania, could appreciate the difficulty of finding suitable paths for flying robots. I would like to thank Dr. Lesieutre for reviewing the thesis and providing helpful intellectual insights.

I will miss everything about our AVIA lab. From morning coffee to night outs in the lab. I would like to take this opportunity to thank Nathan Depenbusch and John Bird. I have not seen any one else apart from them who are more passionate about aircrafts. It was an absolute privilege to visit the Air and Space Museum at DC with them. We worked together on several projects. The long discussions we had about research and other topics has helped me in a lot.

Special mention should also be made of Sean Quinn Marlow, who’s helpful insights have always been there whenever I needed them. I would also like to thank John F. Quindlen, Shane Tierney, Sana Sarfraz, Nicholas Grande, Zeljko Raic, Kwok Cheng, Shawn Daugherty , Zuqun

Li, Ben Truskin, Dmitriy Makovkin and William Holmes for creating an atmosphere in the lab which was both fruitful and enjoyable.

While lab was fun, my friends outside the lab made weekends all play and no work. From long drive on long weekends to midnight ghost hunting on full moon nights, my friends in State College made sure weekends was never boring. While Dr. Supratim Ghosh made sure you need not worry about a vacation plan, Smarajit Mondal and Sravani Banerjee took Indian cuisine to a whole new level. All credit goes to Debanjan Das for the fore-hands I learned in lawn tennis.

I am proud that I was born to parents who have inculcated within me the quest for knowledge the indomitable desire to know. My father's greatest teaching was to view any difficult situation as an opportunity to explore all possibilities; while my mothers teaching of finding faults in yourself rather than everyone else helped to tackle many challenging aspects of grad life and life in general. My brother Ayan has been the big brother who has guided my career throughout my life. Like the leader in a birds flying in V formation, Ayan has made my life much easier just following his wake. Ayan and Rituparna will always be my inspiration.

Perhaps the greatest find of my grad life has been my friend and wife Amrita. Amrita can light up any moment with her smile. I am blessed to have a family so loving and caring. Working together as grad students was both fun and enjoyable. My last year of grad life was most difficult as she graduated and left for Fort Collins. I will wait for the day when she can again join me for the adventures in California.

Dedication

To Amrita

Chapter 1

Introduction

Uninhabited Aerial Vehicles (UAVs) are commonly used in many applications. Be it search and rescue, surveillance, or first responders in a natural calamity UAVs are now an indispensable companion for armed forces and rescuers. UAVs are being used for many applications like coastal surveillance, hurricane watch, traffic control and have the potential to be used for many other applications like precision agriculture, package delivery etc. All of these applications rely on a stable aerodynamic platform and a reliable energy source.

Small UAVs suffer from limited capacity for on board energy storage. Often there is explicit trade off between fuel and sensing payload. Moreover, due to their small size and low typical flying speeds, UAVs operate at low Reynolds number (denoted by Re: it is a dimensionless parameter which defines the ratio of inertial forces to viscous forces.) At low Re, viscous forces becomes very important and the aerodynamic performance of a small UAV is much worse than its large counterpart. Thus, the two-fold effect of reduced aerodynamic performance and incapability of carrying larger payloads, limit the mission capabilities of UAVs. Though improvements of battery technology will enhance their capabilities, immediate performance gains can be obtained by extracting energy from the atmosphere.

Wind has significant contribution to fuel consumption of an aircraft, but the effect of wind can be minimized by computing optimal speeds to fly and by computing optimal routes. Indeed in many cases it is possible to significantly reduce fuel consumption by both flying at optimal speeds and by flying a route that either exploits favorable winds or avoids unfavorable winds.

In typical powered flight operations, cruise flight is done at constant airspeed and power. For propeller-powered aircraft maximum range occurs at the speed for best L/D: however this is only true in calm conditions. It is perhaps intuitively obvious that range is affected by wind;

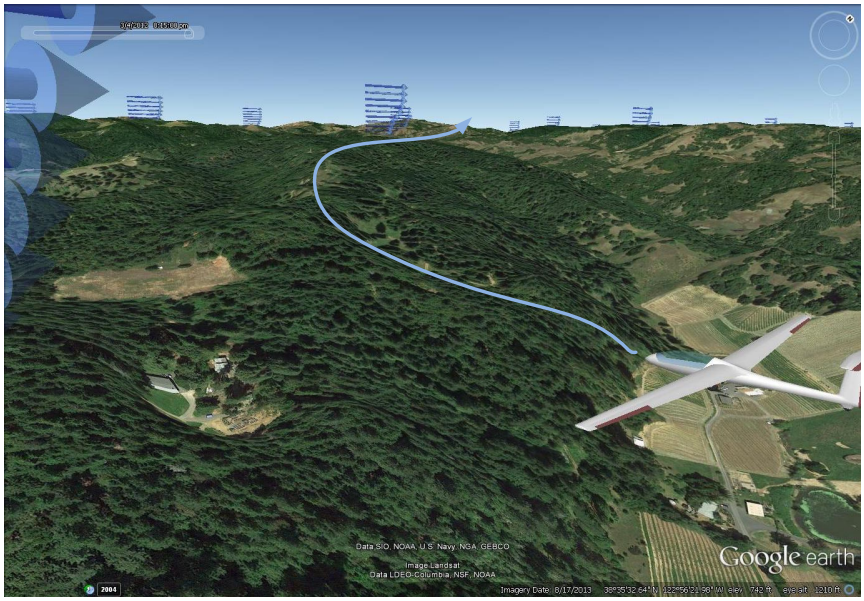


Figure 1.1. Schematic of Mission Scenario: The aircraft is looking to find minimum energy routes through a given wind field to reach a destination.

it is less obvious that the range-maximizing speed is also affected by wind. It should be noted that sailplane pilots are already familiar with this; general aviation pilots perhaps less so.

Route planning can also be used to reduce fuel consumption. Careful trajectory planning to avoid headwinds and rerouting through regions of free lift can significantly increase fuel efficiency. Figure 1.1 shows the schematic representation of the problem. The aircraft is looking to find suitable energy efficient route through a complicated wind field.

The problem at hand is to identify favorable winds and utilize wind to minimize fuel consumption. The planning problem is complicated by the complex nature of the wind field (Figure 1.1 shown as blue arrows). Note the change in magnitude and direction with height above terrain. Temporal variation (not shown in the figure) can also be significant. Optimal routes can be identified through path planning techniques by exploring and exploiting the wind fields which can drastically reduce the fuel consumption of aircraft.

A similar problem arises for ground vehicles navigating through an unknown terrain ((Figure 1.2). Energy consumption of a ground vehicle is determined by the slope of the terrain and terrain type (for example it takes more power to drive up a slope or driving through a muddy region than on flat level surface). For planning for ground vehicle it is assumed that the terrain remains static.

Figure 1.2. Ground vehicle navigating through difficult terrain. Image source: *https://www.peruadventurestours.com*

This research will:

- provide a mathematical framework for energy-aware flight planning through arbitrary wind field;
- sampling based motion planning techniques will be discussed that can be used to harvest energy from the atmosphere;
- discuss optimality and completeness of the planning approach;
- demonstrate the effectiveness of using the methods for energy harvesting using simulation results for a flight vehicle.
- demonstrate the effectiveness of the planner for finding energy efficient routes in difficult terrain for a ground robot.

1.1 Motivation

This research is motivated by the desire to improve energy efficiency of small UAVs. For small UAVs, vehicle speed is comparable to wind speeds. Hence wind has significant contribution towards flight efficiency.

(a) Red-tailed hawk (*Buteo jamaicensis*). This bird is widespread throughout North America. Image source: www.richard-seaman.com.

(b) Gliders using ridge soaring for long distance travel. Image Source: http://en.wikipedia.org/wiki/File:RidgeSrn.gif

Figure 1.3. Practical applications where wind energy is used for long distance travel.

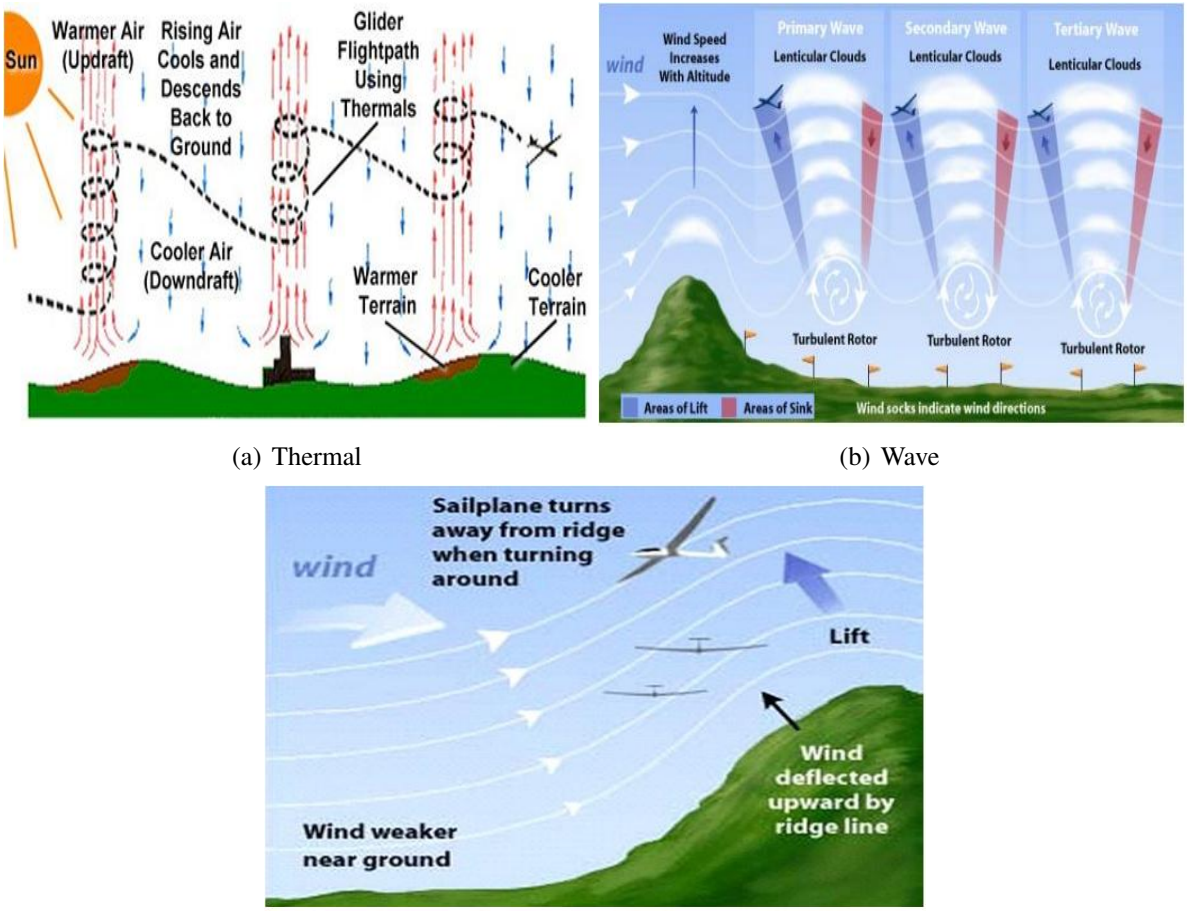
This dissertation describes a framework to enable planning for energy-efficient flight in complex, four dimensional wind field. This framework can also be applied to energy aware motion planning for ground vehicles in complex environments. The flight of birds and sail planes inspired the research presented here, but the foundations of the solution lie in the broad research field of robot motion planning.

Birds constantly use atmospheric energy to minimize energy expended for their travel. Large birds such as hawks (Figure 1.3a) and eagles as well as human sailplane and hang glider pilots (Figure 1.3b) have been exploiting the energy available from updrafts of air to fly for hundreds of kilometers without flapping their wings or the use of engines. Migratory birds routinely follow certain routes while traversing their journey to a distant goal. For these long range flight paths, updrafts are the main source from which energy can be exploited.

Updrafts of winds are caused mainly by the three governing factors (Figure 1.4):

- uneven heating of the ground, which produces buoyant instabilities known as thermals;
- long period oscillations of the atmosphere, generally called wave, which occurs in the lee of large mountain ranges;
- and orographic lift, where wind is deflected by the slopes of hills and mountains.

Typical life spans of updrafts varies from minutes (for thermals) to hours or days (for ridge and wave lift). Ridge lift and wave are predictable phenomenon, and thus these can be used by



(a) Thermal

(b) Wave

(c) Ridge Soaring Image source: http://www.aerospaceweb.org/question/nature/q0253.html

Figure 1.4. Different types of updrafts used for static soaring.

trajectory planning techniques to compute paths which exploit the vertical air motion for long range flights.

A second means of extracting energy from the air is known as dynamic soaring. Dynamic soaring uses velocity gradients (which can occur near the ground due to the boundary layer) or shear layers (which often occur on the leeward side of mountains and ridges). (Velocity gradient is the vertical gradient of the mean horizontal wind speed. It is the rate of increase of wind strength with unit increase in height above ground). This strategy, was first described by Lord Rayleigh in an analysis of albatross flight [7, 8]. Dynamic soaring is again becoming the subject of research both for recreational flight (mainly by RC flying enthusiasts) and for UAV flight. However, this class of dynamic soaring generally requires highly agile flight in close proximity to the ground: this is a very risky endeavor.

The third means of extracting energy from the air exploits gusts. It has been observed that the flight performance of large birds is improved by gusts, while it is typically reduced on human-piloted aircraft [9]. This suggests that birds are able to extract energy from gusts. Kiceniuk has reported that it is even possible to extract energy from a downward gust (a downward gust is a gust with net component of wind in the downward direction.) [10]! Extracting energy from gusts is complicated by their typically short duration, hence very fast response (typically exceeding human reaction time) is required. Control laws have been developed to enable energy extraction from gusts by small UAVs [11].

These three methods of extracting energy from the environment can be used to enable autonomous long duration, long distance (denoted by $(LD)^2$ flight) for unmanned air vehicles. These three methods of extracting energy are referred to as static soaring, dynamic soaring and gust soaring respectively. For long range path planning, static soaring phenomenon is suitable to be exploited based on the time scale of occurrence and amount of energy that can be extracted from it. The focus of this research is on static soaring by an autonomous aircraft.

The major focus of this research is to utilize static soaring for enabling long endurance and long range flight. Paths will be planned according to minimum energy utilization and maximum energy available from the atmosphere. Note that path planning requires at least some knowledge of the environment (or estimates of environmental conditions): here it is assumed that a weather forecasting tool such as MM5 [12] or WRF [13] is available to provide forecasts of winds aloft.

1.2 Related Work and Previous Work

The research presented in this dissertation is motivated by robot path planning and flight of birds. There has been tremendous amount of research relating to the problem of path planning for mobile robots. Robot path planning in a cluttered environment is a widely studied problem. Much of the research presented in this dissertation draws inspiration from nature's solution to path planning in natural environment. This section presents a more detailed discussion in corresponding fields of study.

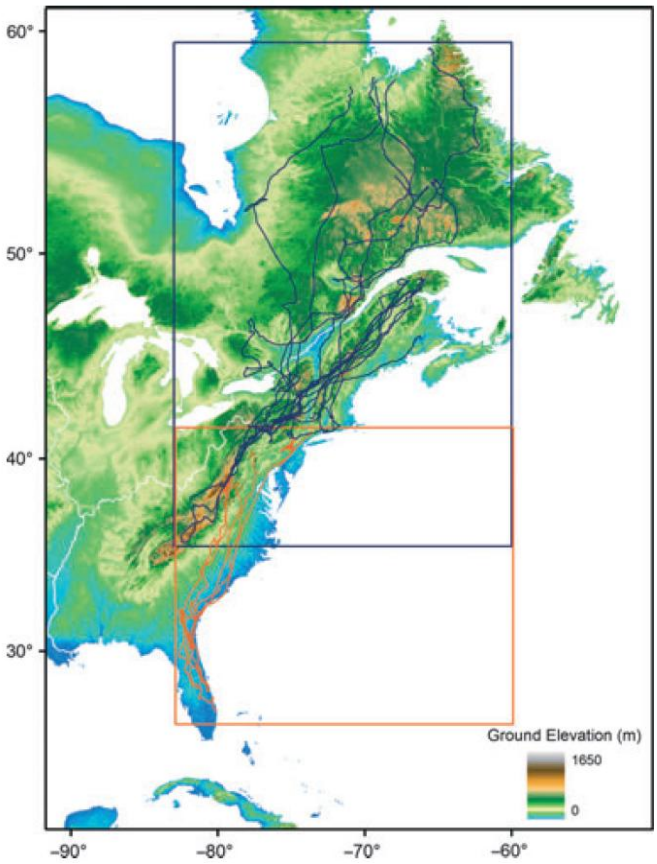


Figure 1.5. Bird Tracks of North America. Turkey vulture in orange; golden eagle in blue. [1].

1.2.1 Flight of Birds

One of the earliest analyses of soaring flight was published by Lord Rayleigh in 1883 [14], where he described the flight of the albatross. The albatross is able to extract energy from the gradient in the wind field from the ocean's surface to about 20m altitude. More recently researchers such as Gottfried Sachs [15] and Colin Pennycuick [16] have examined albatross flight in more detail.

Soaring birds that breed in the northern hemisphere migrate southward in the autumn. These birds include raptors, storks and pelicans. These birds are quite heavy and they depend on flapping flight only for short duration. Hence for their long migratory journeys they rely on energy from the updrafts of the wind. In [1] Bohrer et. al contrasts the migratory strategy of golden eagles and turkey vultures depending on usage of updrafts that can be derived from the Appalachian mountains.

The migration patterns observed from GPS tracked birds have shown that they follow spe-

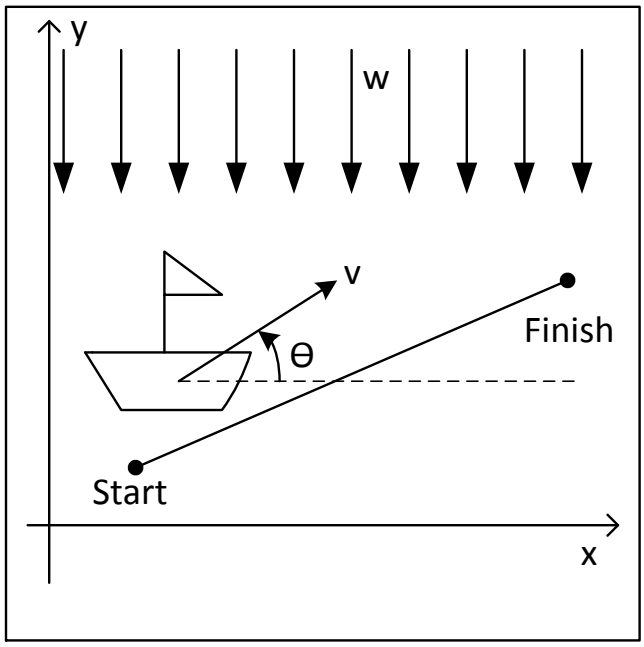


Figure 1.6. The Zermelo Problem.

cific regions in the atmosphere for their migration (Figure 1.5). The goal of this dissertation is to find energy efficient routes for UAVs for long range long endurance missions. To enable small UAVs for long range missions planning in a flow field is imperative.

1.2.2 Path Planning in a Flow field

Path planning of air vehicles in presence of winds has received a considerable amount of attention over a long time. The general problem of planning in a flow field has been studied extensively. In 1931 Zermelo solved the problem of optimal navigation of small ships in presence of currents[17] (Figure 1.6). Modern researchers like Ghose et. al [18] has solved the time optimal problem as a two point boundary value problem using techniques like multiple shooting method.

Early pioneering work for optimal path planning with vehicle constraints was done by Dubins [19]. Dubins' work was based on ground vehicles with no reference to flow field, and showed that time optimal paths consisted of trajectories parameterized as turn-straight-turn. Recently the time optimal problem has been adopted as a Markov Dubin problem as

suggested by Sussmann [20]. In the presence of steady uniform winds the Zermelo-Markov-Dubins (ZMD) problem has been solved by McGee and Hedrick using Maximum Principle of Pontryagin [21] [22]. In [23] Bakolas has also given a time optimal synthesis for the ZMD problem. However in all these cases only steady uniform flow/wind is considered; furthermore only minimum time trajectories are examined.

Path planning for underwater gliders is an emerging field of study. In [24] Rao uses a sampling based method, namely RRT (discussed later) in real ocean currents. In [25] uses an A* planning method to find minimum time paths in Mediterranean sea. In [26] Lermusiaux et. al uses level sets to find minimum time paths in a flow field. In all these cases time minimization is considered and the cost function is *admissible* (i.e. an estimate of the cost-to-go: this is described in more details in Chapter 2). Moreover search based methods are computationally expensive for real time applications.

1.2.3 Autonomous Soaring

A rich and varied literature is present in the field of optimal static soaring trajectories with the application of human-piloted soaring flight [27], [28].

Autonomous static soaring has received a lot of attention in recent years. Simulation results of thermal flight are reported by Allen (2005) [29] and flight test results are presented in Allen (2007) [30]. Edwards reports very impressive results of autonomous thermal soaring [31]. However, these do not consider the problem of trajectory planning.

Wind routing for powered aircraft has been considered for both crewed and uncrewed aircraft. Rubio describes a planning method based on genetic algorithms [32]; Jardin discusses a method based on neighborhood optimal control [33]. Neither of these approaches consider the possibility of harvesting energy from vertical components of the wind field. Several authors have addressed the optimal static soaring trajectory problem in the context of soaring competition. The MacCready problem [34, 35], the final glide problem [36], and "Dolphin" flight along regions of alternating lift and sink [37, 38, 39] all address optimal static soaring including optimal speed to fly between thermals of known strength. de Jong [40] describes a geometric approach to trajectory optimization. Most of this research is limited by known lift distribution (e.g. sinusoidally varying lift [41] or "square wave" lift [42]) and generally do not consider the effects of horizontal wind components nor does it consider temporal variation in wind field.

1.2.4 Graph-based Approaches to Flight Path Planning

Path planning methods is a fundamental problem in robotics. The state space for motion planning is a set of possible transformations that could be applied to the robot. This is referred to as the *configuration space*. The concept of *configuration space* was first introduced by Lozano-Perez and Wesley [43], [44]. The planning in the configuration space is typically approached using one of the three methods: search based, sampling based or a combination of both. Search based methods are most widely used in the field of robotics owing to its ease of implementation. The idea behind search based planning is simply to search a regularly sized grid cell which represents the configuration space.

Cellular decomposition approaches to robot path planning have been widely used in many practical application. The robots configuration space is divided into finite cells and the problem of finding a continuous path between a start and goal is reduced to finding a sequence of adjacent cells (e.g. Stentz [45]). These graph-based techniques have been very successful for obstacle avoidance for many ground based robots as well as for some UAV applications like radar evasion [46].

Several techniques have been developed to compute cost-minimizing paths through a graph. Classical graph search algorithms have been developed for calculating minimum cost paths in a weighted graph. The two most widely used algorithms are Dijkstra's algorithm [47] and A* algorithm [48], [49]. Both the algorithms are special form of dynamic programming[50] and return optimal paths. A* algorithm guides the search towards promising sites with heuristics. Dijkstra's Algorithm, which is a variant of breadth first algorithm, can only handle positive cost. Algorithms like Bellman-Ford which are tailored to handle negative costs encounters problems when negative cycle exists [51]. A* algorithm removes the problem of negative cycle by imposing constraint of allowing node visit only once.

These above approaches work very well when the planning scenario is known a priori. Modern researchers use modified graph based algorithms to implement in real life applications. The most common method to handle unknown environments is to update the graph and re-plan new paths when new information arrives.

Instead of re-planning from scratch, which is computationally expensive, a far more efficient solution can be found by repairing the current solution. A number of solutions [52], [53] exists for efficient re-planning. Focused Dynamic A* (D*) [54] and D* lite [53] are the most widely used algorithms for their efficient use of heuristics and incremental updates. Both D* and D* lite are very similar algorithms which modify only certain sections of the graph which

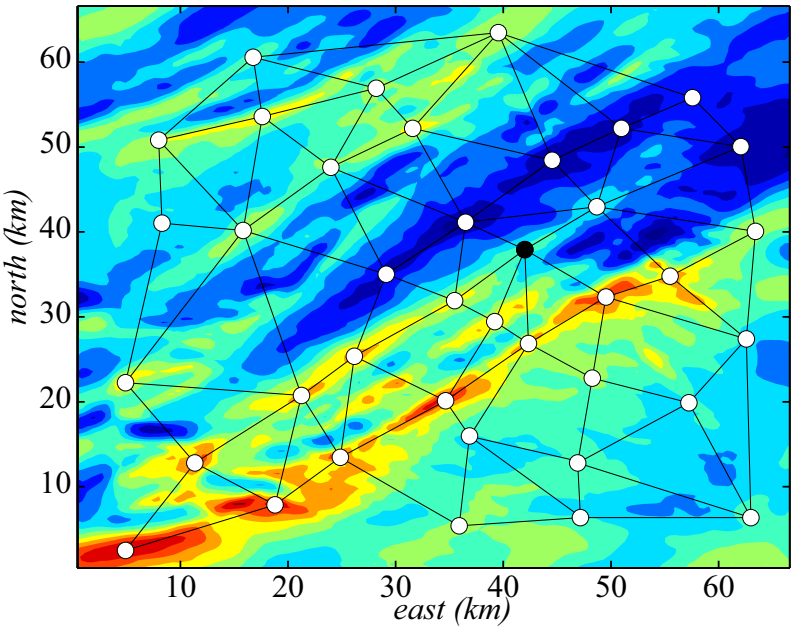


Figure 1.7. Schematic of graph-based planning for autonomous soaring over a wind field.

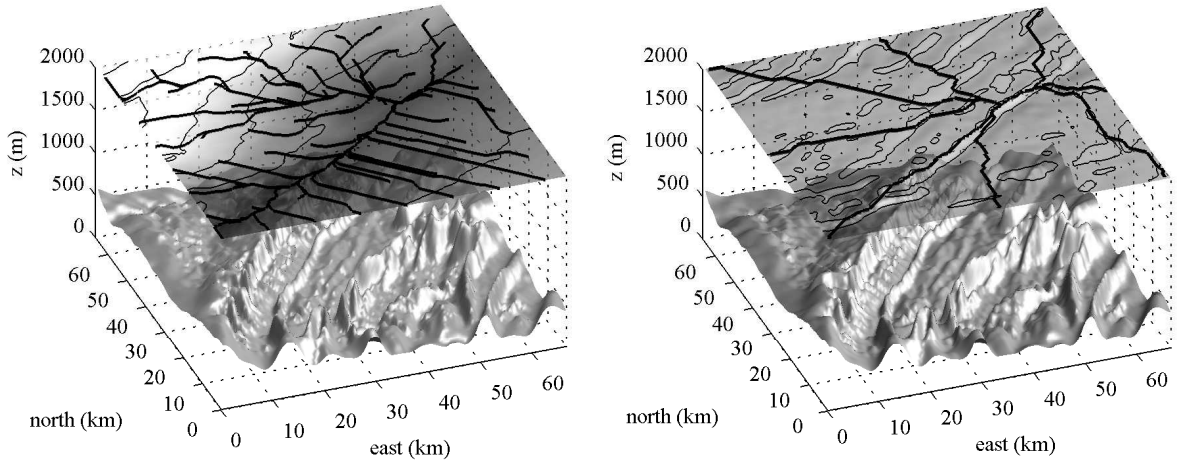
are affected by changing environments. These have been successfully implemented in ground based robots (Pioneers, and E-Gators [55]). For UAV applications however re-planning is difficult owing to constraints of always moving forward and feasibility of path. Moreover planning in 3-D with time varying wind fields the search space increases exponentially.

1.2.4.1 Previous Work

Previous research in this field was based on graph based planning techniques [56]. The continuous space was first discretized by suitable nodes (Figure 1.7). A wave front expansion algorithm called *Energy Map* was used which computes the minimum total energy required to reach the goal from an arbitrary starting point while accounting for the effect of arbitrary wind fields.

The *Energy Map* provides a path to the goal as a sequence of way points, the optimal speeds to fly for each segment between way points and the heading required to fly along a segment. Since the energy map is based on the minimum total energy required to reach the goal it immediately answers the question of existence of a feasible solution for a particular starting point and initial total energy.

The results obtained from energy map were compared with A*. The cost function for the A* algorithm was the weighted sum of energy required and remaining expected energy



(a) Energy map, 2000m altitude

(b) A* paths, 2000m altitude

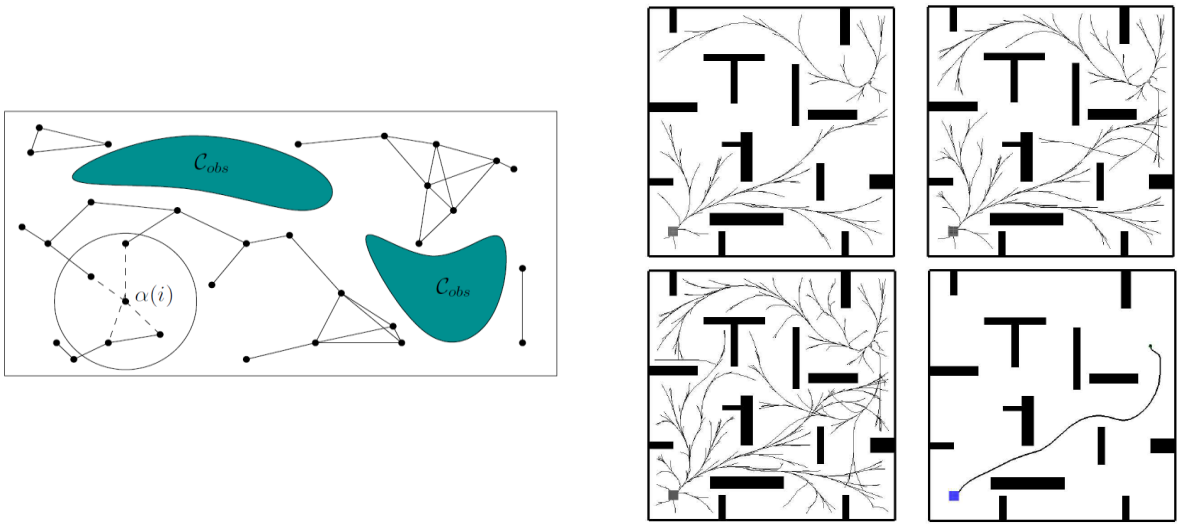
Figure 1.8. Flight paths for energy map and A*. Left side: paths at 1000m altitude; right side: paths at 2000m altitude [2].

consumption to goal. The effect of varying the weight parameter on the flight paths were examined. The energy expended along a path for varying weight is examined, and the results are compared with a wavefront expansion planning algorithm. The weight is selected based on maximum energy utilization that is available from the atmosphere and minimizing time to reach the goal. Optimal weight is selected based on simulation results. Both the methods of path planning are used in real wind field data. Energy efficient routes were found in the real wind field using both the methods.

Figure 1.8 shows the paths discovered by *Energy Map* and A* at 2000 m altitude over Central Pennsylvania [2]. It was shown that the *Energy Map* approach was better than A* approach for finding minimum energy paths in spatially varying wind fields.

Graph based planning techniques can be very useful because of their ease of implementation. When specific way-points are given and wind field in known a priori graph search algorithms can be used to plan and optimize flight plan. The *EnergyMap* described earlier was used in the NASA Green Flight Challenge the details of which are provided in Appendix A

Graph search algorithms work very well in low dimensional problems. Thus for planning in static 2-D wind field, graph search method gave good solution. For 3-D time varying path planning the search space grows exponentially and feasible solutions cannot be found in specified time-bounds. This propels the research towards sampling based solutions for large state space search.



(a) PRM (b) RRT

Figure 1.9. Probabilistic Road Map and Rapidly Exploring Random Tree. Image Source: [3].

1.2.5 Sampling Based Planning

Graph search algorithms are *complete*. Since the entire space is searched for a feasible solution they report a solution if one exists. Complete algorithms suffers from severe computational complexity. A remarkable result was proved by Reif in 1979. Reif showed that the most basic version of motion planning problem, called the piano movers problem was PSPACE-hard[57]. This strongly suggested that complete planners are unsuitable for practical applications.

Practical planners were introduced with the development of sampling based motion planners [3]. These approaches relaxed the completeness requirement to, for instance, resolution completeness or probabilistic completeness. Resolution completeness guarantees completeness only when the resolution parameter of the algorithm is set fine enough. Probabilistic completeness ensures asymptotic completeness as the number of random samples approaches infinity. These planners demonstrated remarkable performance in accomplishing various tasks in complex environments within reasonable time bounds .

One of the very early sampling based motion planner is that of Kondo[58]. Kondo's planner strongly reflects classical grid search approach where the planner searches a fine grid placed over configuration space. In 1990 Barraquand and Latombe introduced the Randomized Path Planner (RPP) [59]. RPP planner defines several potential fields over a grid imposed on the workspace. Each potential field corresponds to a "control" point on the robot. The plan-

ner descends the gradient of the configuration space and uses random walks to escape local minimum. This works iteratively until the goal is reached. In 1990 Glavina introduced the ZZ-method [60]. The ZZ-method attempts to connect start and goal points through "straight and slide" local planner. New configurations which are subset of the entire space are chosen until a path to the goal can be found.

Probabilistic Roadmaps [61] and Rapidly exploring Random Trees (RRT) [62] are widely successful in solving path planning problems in static and dynamic environments. Probabilistic road-maps is a multiple query planner where initially the free space is sampled and a local planner is used to connect the sampled points. In the second state the connected graph is searched via Dijkstra's algorithm. RRT algorithm on the other hand is a single query planner where the roadmap is build incrementally along with sampling. Even though these algorithms do not achieve completeness, they provide probabilistic completeness guarantees in the sense that the probability that the planner fails to return a solution, if one exists, decays to zero as the number of samples approaches infinity.

1.2.5.1 Sampling based planners with cost

Most of the research in sampling based motion planners is concerned with obstacle avoidance in high dimensional space. The quality of the results produced like optimality of paths has mostly been ignored. Recent research in this field has been done by Jailiet et al [63]. In this research RRT planners are based on cost of transition over terrain and the planner picks lower energy paths for transition. No claims of optimality has been addressed. Recently an extension of RRT algorithm called the RRT^* [4] has been shown to be an optimal motion planner in static environments. However RRT^* has not been implemented in time varying environments and cannot be used to handle non-admissible cost. It will be shown in Chapter 2 that energy harvesting flight paths result in non-admissible costs.

This research focuses on the problem of trajectory planning in non-uniform, unsteady wind fields. Because of the significant spatial and temporal variation in the wind field the state space of the vehicle is very large, leading to a computationally intensive trajectory planning problem. Sampling based path planning techniques is widely successful in solving this "curse of dimensionality". Finding an optimal trajectory in such an complex scenario is extremely challenging. A new sampling based motion planner is proposed to handle time varying wind fields.

1.3 Overview of Research

Long Range path planning in time varying complex wind field is an extremely challenging problem. To find optimal paths both time and space complexities have to be considered. Feasibility of paths will depend on changing environmental conditions and thus paths feasible at a certain point of time may not be feasible at any other time. This is an extremely challenging optimization problem. Finding energy efficient routes for a ground vehicle in mountainous terrain poses similar optimization problems but with static time.

Since sampling based methods are most suitable to handle high dimensional problems, sampling based planning is considered in this research. Time has to be incorporated explicitly to handle time varying winds. This research introduces a sampling based method named the *Kinematic Tree*. The Kinematic tree is an incremental sampling based motion planner based on "survival of the fittest". Time is explicitly taken into account in the planner as an optimal path is function of both time and space. Time of start is extremely crucial for feasibility of paths and thus ways to incorporate delayed launch in the planner itself is considered. For all the planning problems discussed in this research it is assumed that the wind information is known a priori.

Planning is based on kinematic model of the vehicle, since it is most suitable for implementation. It is assumed that the vehicle is in trimmed condition between waypoints planned and the time taken to travel between waypoints is sufficiently large compared to response time of the controller. Hence a kinematic model is sufficient for planning.

To improve computational time a set of branches (computed waypoints) is pre-computed from the space of allowable inputs allowing fast expansion of nodes. The kinematic tree is shown to be resolution complete. Resolution complete means that if a path exists then a tree of sufficient resolution will find it: this is analogous to probabilistic completeness of randomized motion planners. Energy cost is non-admissible. Hence global optimality of the cost function cannot be guaranteed for this planner. But if the cost function is admissible, for e.g say the cost be minimizing distance which is an admissible cost function, the planner is shown to be optimal upto resolution. The planner is shown to be computationally efficient.

To improve computational time, a set of *branches* is pre-computed. A branch in a tree refers to the computed waypoints based on allowable inputs. Set of branches in the tree lays the waypoints to reach a destination. The planner is shown to be computationally efficient.

The Kinematic Tree algorithm is shown to be *resolution complete*. Resolution complete

means that if a path exists then a tree of sufficient resolution will find it: this is analogous to probabilistic completeness of randomized motion planners.

It will be shown in the next chapters the energy cost is non-admissible. Hence global optimality of the cost function cannot be guaranteed for this planner. But if the cost function is admissible, for e.g say the cost be minimizing distance which is an admissible cost function, the planner is shown to be optimal upto resolution.

Issues relating to global optimality is covered in details in this dissertation. Often there is explicit trade between optimality and computational complexity. The same happens for the optimal version of Kinematic Tree algorithm named the Kinematic Tree* (KT*). This dissertation will present a probabilistic analysis of average complexity of KT* algorithm so that it can provide feasible suboptimal paths in realistic time bounds.

The Kinematic Tree algorithm is tested for real wind data over central Pennsylvania. Wind data is obtained from the Meteorological department at Penn State. Flight to a distant goal is considered for a glider which can only reach the goal by exploiting energy from the atmosphere.

The Kinematic Tree* algorithm is explored for path planning of a ground robot in difficult terrain. The algorithm is tested to find optimal paths in mountainous terrain where the cost of transition is highly dependent on traversibility of the terrain.

1.4 Contributions

The major contribution of the dissertation are outlined below.

- **Sampling based planning framework for exploiting wind** A new sampling based planning method called the *Kinematic Tree* was developed to effectively handle spatial- and temporal varying wind fields.

- **Theoretical insights on the planner** Computational complexity, completeness and optimality of the proposed planner are discussed in lengths and compared with standard methods.

- **Routes of minimum energy in a real wind data** Simulation results show the efficacy of the proposed method in a time varying complex wind.

- **Application of the method for ground robots in difficult terrain** The optimal *Kinematic Tree* algorithm named the *Kinematic Tree** was used to find minimum energy

paths for a ground robot in mountainous terrain.

1.5 Readers Guide

The remainder of the dissertation is organized is follows

- **Chapter 2** begins with a brief discussion of casting the problem of finding a path in a complex wind field as a sampling based motion planning problem. It then discusses the problems of using existing algorithms which motivates the need for a new algorithm.

- **Chapter 3** introduces the Kinematic Tree algorithm. Some of the properties of the planner like the computational complexity, completeness and conditions for optimality are included. In-depth discussion about the effect of heuristic on computational complexity is considered.

- **Chapter 4** describes how to create a Kinematic Tree from a kinematic model of an aircraft and shows the simulation results of the Kinematic Tree path planner over wind data over Pennsylvania. Energy efficient routes are determined in a given wind field.

- **Chapter 5** presents the simulation results of optimal version of the Kinematic Tree algorithm named the Kinematic Tree*(KT*). The KT* algorithm is shown to find optimal paths while avoiding obstacles. Planning for minimum energy routes for a ground robot in mountainous terrain is also shown.

- **Chapter 6** summarizes the results of this research and presents the concluding remarks and future works.

Chapter 2

Sampling Based Motion Planning for Energy Aware Flight

This chapter defines the path planning problem in time varying wind fields. It has three purposes: (a) set up the path planning problem as a sampling based planning problem; (b) discuss the merits and demerits of existing algorithms for planning in a flow field ; (c) provide some justification as why none of the existing algorithms can be used directly to plan paths in this scenario. Most of research in sampling based planning is concerned with obstacle avoidance. The general framework of sampling based motion planners is adopted for finding optimal paths in a flow field. Chapter 3 describes the proposed sampling based algorithm named the *Kinematic Tree* algorithm and Chapter 4 shows the Kinematic Tree algorithm does indeed find feasible paths in a time varying wind field.

The main idea behind using sampling based motion planning technique is avoid explicit construction of the entire search space for a feasible solution. Graph based methods discretize the entire space using regular grid cells and uses search methods to find a feasible solution. Sampling based methods, in contrast, discretize the environment by picking random samples and uses different methods to find a path. Thus there is weaker guarantee of *completeness*. An algorithm is said to be *complete*, if, for any input it reports correctly whether there is a feasible solution. Weaker notion of completeness such as probabilistic complete or resolution complete are introduced for sampling based methods. Probabilistic completeness of a planner, refers to the probability of finding a path approaches one as the number of samples approach infinity. Resolution completeness on the other hands guarantees completeness if the resolution parameter (of either control or density of samples) is tuned fine enough.

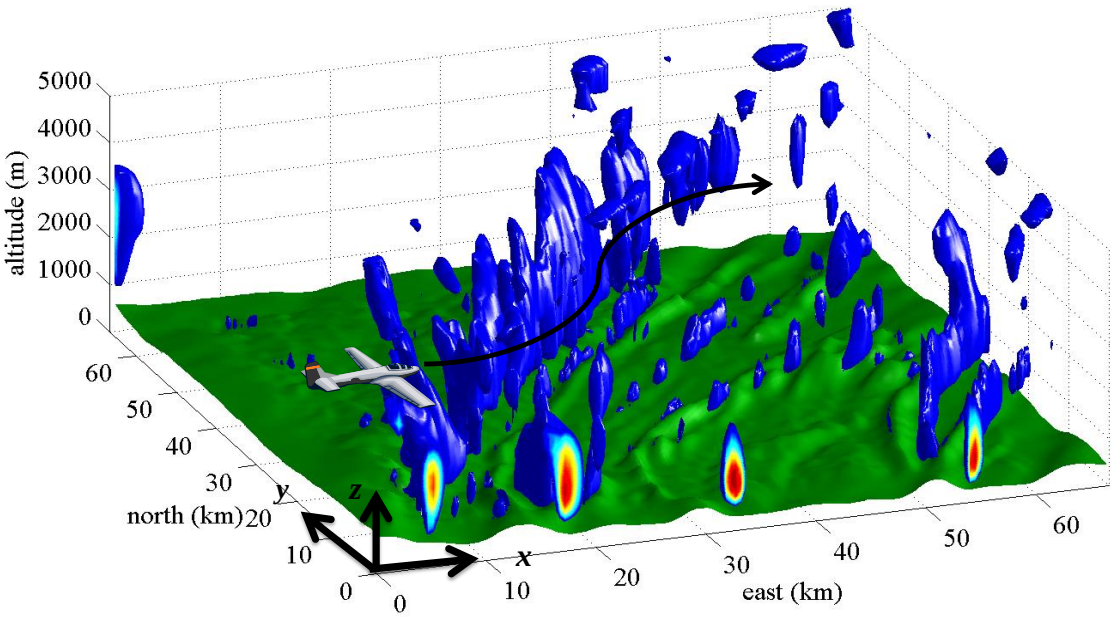


Figure 2.1. Schematic of Mission Scenario.

To provide some intuition into the difficulties associated with planning in time varying wind field this chapter includes a brief discussion of existing methods: Probabilistic Roadmaps, Rapidly Exploring Random Trees(RRT) and RRT*. This motivates the design of a new planner the Kinematic Tree.

Section 2.1 defines the planning problem and the how the specific problem of minimizing energy in a time varying wind field can be cast as a sampling based planning problem. Section 2.2 defines the general framework for sampling based methods to solve a path planning problem. Section 2.3 discusses the existing algorithms and finally Section 2.4 provides a summary.

2.1 Problem Formulation

A plan in general refers to a strategy or behavior of a decision maker. A plan specifies a sequence of actions to be taken to achieve an objective. Planning problems involves a *state space* that captures all possible situations. For aircraft the state refers to its position, orientation and velocity. For time varying problems time can also be considered as a *state* (although it is uncontrollable). A plan generates a set of inputs or controls to be applied to the robot at specified

time.

The problem is to compute an energy optimal trajectory through a complex, time varying wind field (as the one shown in Figure 2.1). Here "complex" means significant spatial and temporal variation over the scale of the resulting flight plan. Given a prediction of wind field, the problem at hand is to compute a set of waypoints defining the flight path that minimizes the energy required to reach the goal. The blue regions in Figure 2.1) are potential regions from which lift can be derived from the environment (See Chapter 4 for more details on the wind field). Thus a plan which can exploit free lift from the atmosphere is the goal of this research.

Path planning under differential constraint can be considered as variant of classical $two-point\ boundary\ value\ problems$ (BVPs). In that setting, initial and goal states are given and the task is to compute a path through the state space that connects the initial and goal states while satisfying path constraints.

It is assumed that the differential constraints are expressed in a state transition equation $\dot{x} = f(x,u)$ on a metric space X called the $state\ space$ (these constraints represent a dynamic or kinematic model of the vehicle). A solution path is derived from an action trajectory via integration of state transition equation.

The general problem of path planning of a robot moving in an obstacle field can be described as a search problem in metric space (state space) X. Let X represent the entire configuration space of the robot. Let X_{obs} represent the obstacle space. Then the free space can be defined as $X_{free} = X/X_{obs}$. The goal is to find a continuous path from an initial position x_{init} to a goal region $X_{goal} \subset X$ while avoiding obstacles. It is assumed that the initial and goal regions belong to obstacle free regions:

$x_{init} \in X_{free}$

and $x_{goal} \in X_{free}$.

Mathematically the problem can be described as: find a path σ such that $\sigma(t_0) = x_{init}$ and $\sigma(t_f) = x_{goal}$ and $\sigma(t) \cap X_{obs} = \phi, \forall t$.

The optimality problem can be described as: given a planning problem of finding a path from $x_{int} \in X_{free}$ and a goal region $X_{goal} \in X_{free}$ and let a cost be assigned to each waypoint such that $c : \Sigma \leftarrow R^+$,

Then find a path σ^* such that $c(\sigma^*) = min(c(\sigma))$ for all σ.

The specific problem of planning in a time varying wind field can be cast as the general robotic path planning problem, where X_{obs} represents the ground and the cost function associ-

ated with the path is function of wind and control at that point in space and time.

$$c(\sigma) = f(u(t), \mathbf{w}(t)) \tag{2.1}$$

where $\mathbf{w} = [w_x, w_y, w_z]$ is the three dimensional wind field .

2.1.1 Planning with Cost

The optimality problem described in the previous section assumes a cost c associated with the path σ. Most planning problems considers a fixed cost between the planned waypoints. For ground based robots distance covered or time taken is most commonly taken as the cost function. Other cost functions commonly considered are risk of mission failure, minimizing control effort etc.

In aircraft applications total energy can be a critical parameter in trajectory planning (for example, when considering the fuel required to reach the goal). Both environmental and control parameters can affect the energy required for a particular transition: a head wind will increase the required total energy, as will flying at non-optimal airspeed. Thus any sampling based planning technique will require a means of accounting for environmental and control conditions in the analysis of costs of transitions between waypoints.

2.1.2 Planning with Power Ups

In a general wind field vertical winds can be present in addition to horizontal winds. These vertical components of wind can result in a gain in total energy. This is analogous to a "power up" found in video games such as Mario World. These power ups cause many path planning algorithms to fail because they result in *inadmissible* cost functions. Section 3.2.1 describes in details the necessary conditions for an admissible cost function.

In general robotic path planning problems consider cases where the remaining cost to reach the goal can be estimated without over-predicting the cost. This results in *admissible* cost functions. Examples include distance remaining or time remaining to reach the goal. Availability of energy from the atmosphere makes the energy expended by the vehicle to be negative, and this results in an inadmissible cost function.

Classical graph search planning algorithms like Dijkstra's algorithm fails to find a solution if there are negative costs. Algorithms like the Bellman-Ford which are tailored for negative

costs are limited by negative cycles. A* algorithm also fails to generate an optimal path because the estimate of cost to go that is used to compute the total path cost, can over-predict the actual cost to go. In case of randomized planner RRT algorithm has been proved to non-optimal and the optimal version RRT* is optimal in case of admissible cost.

The Kinematic Tree algorithm introduced in the next chapter addresses these problems. Because of the non-admissibility of the cost function global $optimality$ cannot be guaranteed, but the Kinematic Tree algorithm guarantees $feasibility$ of path, if there is a path to the goal. A variant of the Kinematic Tree algorithm, named $KinematicTree^*$ also addresses optimality by considering motion planning problem that has only non-negative cost, such as time to reach the goal (for a mobile ground robot).

2.2 General Framework for Sampling based Methods

A sampling based planning framework will be explored to solve the problem above. Let X represent the entire state space such that initial and goal regions belong to the state space. Let X_{obs} represent the obstacle space and $X_{free} = X/X_{obs}$ represent the free space such that $x_{init} \in X_{free}$ and $x_{goal} \in X_{free}$ The general framework (as described in [3]) for sampling based motion planners cab be summarized as:

1. **Initialization:** Let $G(V,E)$ represent an undirected search graph for which the vertex set V contains a vertex for x_i and possible other states in X_{free} and edge set E is empty. the graph can be interpreted as a topological graph with reachable states $S(G)$

2. **Vertex Selection Method:** Choose a vertex $x_{new} \in S(G)$ for expansion

3. **Local Planning Method:** Generate a motion primitive $u : [0,t_f] \rightarrow X_{free}$ such that $u(0) = x_{new}$ *and* $u(t_f) = x_r$ for some $x_r \in X_{free}$. Check for feasibility of x_r

4. **Insert edge in Graph:** Insert u into E and x_r into V.

5. **Check for solution:** Check whether G encodes a solution path. For sampling based planners thus a goal region $X_G \in X_{free}$ is defined to check feasibility of solution.

6. **Return to Step 2:** Iterate unless a solution is found or some other parameter satisfied. This may return failure in bounded time.

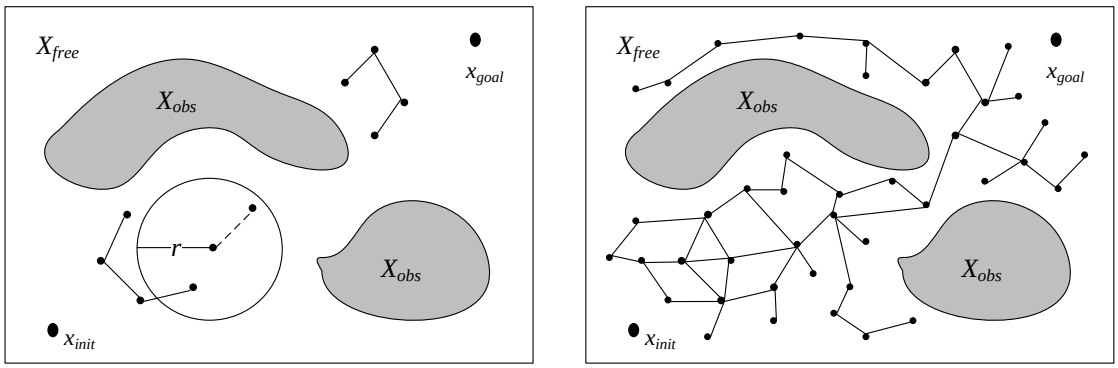


Figure 2.2. Probabilistic Roadmap: Growth of the connected components in the free space

This general framework is used in all sampling based motion planners. The vertex selection procedure can have significant effect on the success of sampling based motion planners. Local planning methods vary between different planners and have significance contribution to computational complexity.

2.3 Existing Algorithms

Some of the sampling based algorithms which are widely used in many practical application are outlined. The difficulty of using existing algorithms for soaring applications is also discussed. All the algorithms follow the general framework described in Section 2.2 and return a graph $G = (V, E)$ where $V \subset X_{free}$. The solution of the path planning problem can be computed from such a graph.

2.3.1 Probabilistic Roadmaps (PRM)

The Probabilistic Roadmaps algorithm [61] is a multi-query algorithm. In its basic version, it consists of a pre-processing phase, in which a roadmap is constructed by attempting connections among n randomly sampled points in X_{free}, and a query phase in which paths connecting initial (x_{init}) and final (x_{goal}) points are sought.

The pre-processing phase is outlined in Algorithm 1 and Figure 2.2 shows the growth of the connected components. The algorithm starts with an empty graph G. At each iteration, a random point $x_{rand} \in X_{free}$ is sampled and added to the vertex set V. Then, connections are attempted between the new sampled point x_{rand} and other existing vercites in the Graph.

Algorithm 1: PRM Algorithm

Function :PRM(x_{init});
$G = (V, E)$;
$V \leftarrow \phi$;
$E \leftarrow \phi$;
for *i=1 to K* **do**
  $x_{rand} \leftarrow random\_config(X_{free})$;
  $U \leftarrow Near(G = (V, E), x_{rand}, r)$;
  $V \leftarrow V \cup x_{rand}$
  for each $u \in U, in\ order\ of\ increasing\ ||u - x_{rand||}$ **do**
    if x_{rand} *and u are not in the same connected component of* $G = (V, E)$ **then**
      if $collision\_free\_path(x_{rand}, u)$ **then**
        $E \leftarrow E \cup \{(x_{rand}, u), (u, x_{rand})\}$
      end
    end
  end
end
Return $G = (V, E)$

Vertices are searched within a ball of radius r centered at x_{rand}, in order of increasing distance from x_{rand}. A simple local planner (like straight line distances) are used to establish connection. Collision free edges are added to the edge set E. To simplify computation connection between x_{rand} and other connected components are avoided. Thus the new point is connected to only the nearest point in the connected graph.

The connected graph $G = (V, E)$ is searched in the query phase to find shortest distance in the connected graph from start to goal. The PRM algorithm is widely used for motion planning. A"simplified" version called the sPRM is used for practical application where connections are established without checking for connected components in the graph.

The PRM thus constructs a connected graph in the free space and finds a solution through the connected graph. Since random samples are drawn from the free space to construct the connected graph there is no guarantee of optimality of the solution found. Indeed in [4] it was shown that the probability that the PRM will find a non optimal path approaches one.

2.3.2 Rapidly-exploring Random Trees (RRT)

One of the most notable sampling based algorithm being used in many applications is the Rapidly Exploring Random Trees[62]. The Rapidly-exploring Random Trees (RRT) combines the two processes of PRM algorithm into a single query process. In its basic version, the algorithm incrementally builds upon a tree of feasible trajectories dictated by a local planner, rooted at the initial position. The resolution complete RRT is a variant of RRT which uses a predefined control input.

Algorithm 2: RC-RRT Algorithm

Function :RRT(x_{init});
G.init(x_{init});
for *i=1 to K* **do**
    $x_{rand} \leftarrow random\_config(X_{free})$;
    Extend(G,x_{rand})
end
Return
Function :Extend(G,x_{rand});
$x_{near} \leftarrow nearest\_neighbour(G, X_{rand})$;
$u \leftarrow select\_input(x_{rand}, x_{free})$;
$x_{new} \leftarrow new\_state(x_{near}, u \in U, \Delta t)$;
if *collision_free_path* **then**
    $G.add\_node(x_{new})$;
    $G.add\_edges(x_{near}, x_{new}, u))$
end
Return G

2.3.2.1 Resolution Complete RRT (RC-RRT)

The RC-RRT algorithm is a discretized version of the the general RRT algorithm. The RC-RRT algorithm is presented in Algorithm 2.

The RC-RRT algorithm incrementally builds upon a tree $G = (V, E)$ defined by its set of vertices V and edges E. A random configuration x_{rand} is picked from the robot's free configuration space (X_{free}). The vertex that is closest to the random configuration x_{near} is chosen for expansion. Then all the available control inputs are checked to see which control brings the robot closest to the random configuration. Most algorithms for practical application uses Euclidean distance to find the closest vertex. If the path joining them is collision free then that

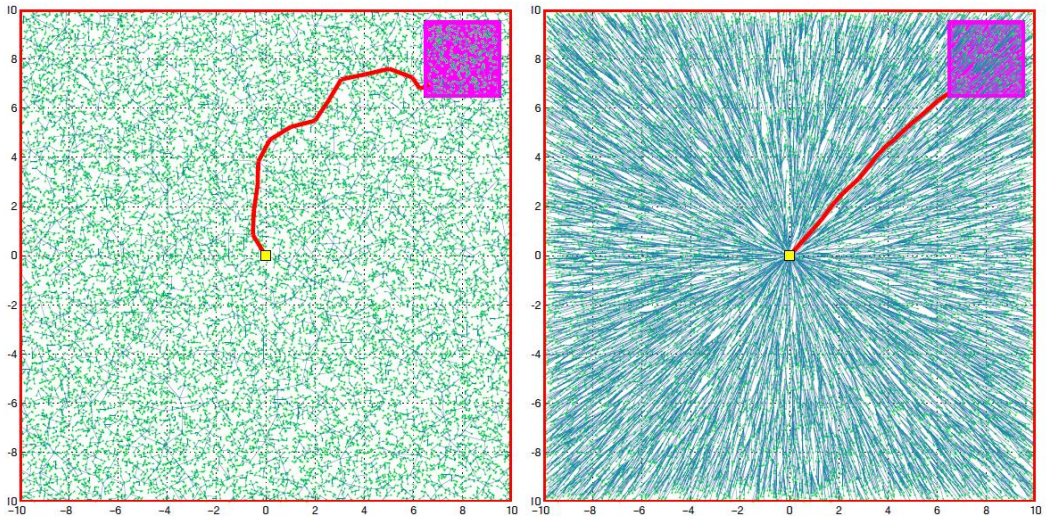


Figure 2.3. Comparison between RRT and RRT* [4].

new point x_{new} is added to the vertex set V and the edge joining the points is added to the edge set E. This process is repeated until the goal is reached.

Uniform sampling procedure of the free configuration space ensures even coverage of the state space. RRT is widely successful in solving many basic motion planning problem and is used for many practical planners.

The resolution complete rapidly exploring random tree (RC-RRT) was introduced by Chang et al. in [64] and was shown to be resolution complete in [65]. However the RC-RRT has not been used in time varying cases and optimality of RC-RRT has not been explored. Recently a variation of RRT algorithm denoted by RRT* has been proved to be optimal for admissible cost functions[4].

2.3.2.2 RRT*

The RRT algorithm has been shown to find feasible paths in different applications. However, one of the main results of [4] conclude that under mild technical conditions, the cost of the solution returned by RRT converges almost surely to a non-optimal value. In [4] Karaman et al. introduced the optimal version of RRT algorithm named the RRT*. The RRT * algorithm has been proven to be optimal.

The RRT* algorithm is shown in Algorithm 3. The RRT* algorithm adds points to the

Algorithm 3: RRT* Algorithm

Function :RRT(x_{init});
G.init(x_{init});
for *i=1 to K* **do**
  $x_{rand} \leftarrow random\_config(X_{free})$;
  Extend(G,x_{rand})
end
Return
Function :Extend(G,x_{rand});
$x_{nearest} \leftarrow nearest\_neighbour(G, X_{rand})$;
$u \leftarrow select\_input(x_{rand}, x_{free})$;
$x_{new} \leftarrow new\_state(x_{near}, u \in U, \Delta t)$;
if *collision_free_path* **then**
  $X_{near} \leftarrow Near(G = (V,E), x_{new}, min\{\gamma RRT * (log(card(V))/card(V))^{1/d}, \eta\})$;
  $V \leftarrow V \cup \{x_{new}\}$;
  $x_{min} \leftarrow x_{nearest}; c_{min} \leftarrow Cost(x_{nearest}) + c(Line(x_{nearest}, x_{new}))$;
  for each $\mathbf{x_{near}} \in \mathbf{X_{near}}$ **do**
    if $Collision\_free(x_{near}, x_{new}) \cap Cost(x_{near} + c(Line(x_{near}, x_{new})) < c_{min}$ **then**
      $x_{min} \leftarrow x_{near}; c_{min} \leftarrow Cost(x_{near} + c(Line(x_{near}, x_{new}))$
    end
  end
  $E \leftarrow E \cup \{x_{min}, x_{new}\}$;
  for each$\mathbf{x_{near}} \in \mathbf{X_{near}}$ **do**
    if $Collision\_free(x_{new}, x_{near}) \cap Cost(x_{new} + c(Line(x_{new}, x_{near})) < Cost(x_{near})$ **then**
      $x_{parent} \leftarrow Parent(x_{near})$;
      $E \leftarrow (E\ \{(x_{parent}, x_{near})\} \cup \{(x_{new}, x_{near})\}$;
    end
  end
end
Return G= (V,E)

vertex set V in the same way as RRT but the RRT* algorithm incrementally modifies a path discovered by an RRT algorithm (Figure 2.3). It considers connections from the new vertex x_{new} to vertices in X_{near}, i.e other vertices that are with a ball of radius $r(\text{card}(V)) = \min\{\gamma RRT * (\log(\text{card}(v))/\text{card}(V)^{1/d}), \eta\}$ from x_{new}. An additive cost function is assigned to each node such that $Cost(v) = Cost(Parent(v)) + c(Line(Parent(v), v))$, i.e, the cost of traveling to a node is sum of the cost to get to its parent plus the cost of transition from the parent node to the current node. New edges are introduced in the tree only based on minimizing the

cost function. If any vertex in X_{near} incurs a cost less than the current x_{new} that vertex is added. The tree is reconnected to maintain the tree structure. The radius chosen is based on an analysis which can be found in details in [4].

RRT* cannot be used for soaring applications mainly because of two reasons: (1) the additive cost function limits the RRT* algorithm for use in admissible cost function; (2) the rewiring step in the RRT* algorithm makes it computationally challenging for use in time varying applications. Details about problems associated with time varying problems are summarized in the next section.

2.3.3 Time-Varying problems

In all the algorithms discussed so far time has not been explicitly considered. In effect, it is assumed that the environment remains static through out the planning period. However, naturally occurring phenomena such as wind, can (and do) vary on time scales that are comparable with the time required for flight of aircraft. Thus planning methods for sampling based approach should incorporate time.

The state space X for time varying problems is defined as $X = C \times T$ where C is the configuration space of the robot and T is time. A state x is represented as $x = (z,t)$ to indicate configuration z and time t of the state vector. Planning in X occurs with only one restriction: that time always moves forward at a fixed, continuous rate.

Many sampling based can be adopted to be used in time varying problems with additional constraints, but this often adds significant complications. For *roadmap* approaches the notion of *directed roadmap* is needed, in which every edge must be directed to yield a time-monotonic path. For each pair of states $(z_1,t_1),(z_2,t_2)$ only one possible edge is possible and the resulting graph becomes a directed graph.

For RRT approach allowable transitions are limited by time constraints. Samples drawn at random have to be checked for feasibility. Samples have to be time stamped and disallowed every time $t_2 \leq t_1$. Bi-directional RRTs which are used in many applications are difficult to implement. Though mathematically it is feasible to have directed graphs in the opposite direction, but starting time is ambiguous for bi-directional trees.

A similar problem arises for RRT* algorithm. The RRT* algorithm modifies paths found by RRT algorithm rewiring the tree to find updated parents of the current node. This leads to a backward propagation in time and for time varying cases this may lead to significant

computational complexity.

2.4 Summary: The Planning Problem

To find a feasible plan for an unmanned aerial vehicle in a real environment a time varying planning framework has to be considered. Wind conditions can vary significantly thought the day. Paths feasible at particular starting time may not be possible at other times of the day due to different environmental conditions. Thus time varying nature of the wind field has to be considered. Exhaustive search of the entire space is not a feasible solution for all practical purposes.

Sampling based methods can be used to find feasible solutions. The general framework for sampling based method will be adopted to find feasible solution in a time varying wind field. But one must have explicit representation of time in sampling based methods to successfully solve the planning problem in a real wind field.

The general framework of goal based sampling based methods, where samples are drawn from free space, can be adopted to draw samples from favorable regions. This leads to an inverse problem of finding controls to get to that position given the constraints of vehicle kinematic and environmental conditions. Moreover the good regions may not be a connected components and connection between good regions may not be established by drawing random samples from favorable regions.

Moreover, planning feasible paths in a time varying environment is an extremely complex planning problem. Feasible paths may include crossing over unfavorable regions of space and getting to regions where there is significant energy from the atmosphere. The entire space cannot be divided binarily into obstacle or free space. It may be possible that the only feasible path passes through a region of unfavorable wind. It way also be possible to have very good regions leading to dead ends. Thus there is need for a planning algorithm to keep track of all the feasible paths time stamped.

This leads to the new planning framework proposed in this dissertation called the $Kinematic\ Tree$. The Kinematic Tree will construct a connected graph $G = (V,E)$ based on all inputs $u \in U$. and keep track of each node at each time step. Though the incremental search will be based on "survival of the fittest", but the dead ends will not make the search to fail owing to the survival of less attractive branches, which may eventually end up to give a solution. The planning framework is discussed in details in the next chapter.

Chapter 3

The Kinematic Tree

This chapter describes in detail the proposed *Kinematic Tree* Algorithm. The Kinematic Tree algorithm was designed specifically for planning in time varying complex environments. Keeping the sampling based framework for path planning, a new algorithm is proposed where time is inherently embedded in the growth of the tree. The Kinematic Tree algorithm is shown to handle 3-D and time varying winds and find feasible paths for an UAV in a wave data set in the next chapter. This chapter presents: (a) The Kinematic Tree algorithm as a sampling based algorithm; (b) proof of resolution completeness, and analysis of computational complexity; (c) extend the Kinematic Tree algorithm for finding optimal paths and prove optimality and computational complexity of the new planner named the *Kinematic Tree**.

The sampling based planning framework was described in the previous chapter. Using the same framework the Kinematic tree algorithm will be defined.

Some of the key issues regarding the completeness and complexity of the algorithm is analyzed. Conditions under which the planner can be made optimal is discussed. The optimal version of the planner, named the Kinematic Tree*, is developed and its properties are also explored. Theoretical results developed in this chapter will be verified through simulation in the next chapters for both the Kinematic Tree and the Kinematic Tree* algorithms.

Section 3.1 described the Kinematic Tree Algorithm. The Kinematic Tree algorithm builds upon a tree structure based on allowable states of a vehicle. The planner will be shown to be resolution complete and computationally efficient. Section 3.2 extends the Kinematic Tree algorithm to its optimal version called the Kinematic Tree* and it will be proved to be optimal. Section 3.3 will discuss in details the trade off between optimality and computational complexity.

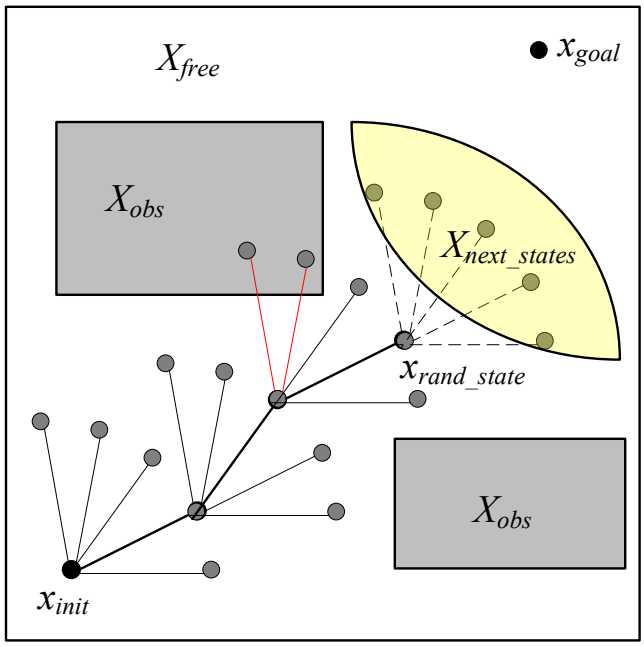


Figure 3.1. Growth of Kinematic Tree

3.1 Kinematic Tree Algorithm

As stated in Chapter 2, let X represent the entire state space of the robot. Let X_{obs} represent the obstacle space, thus $X_{free} = X/Xobs$ represent the free space. Let σ represent a path from the start to the goal. σ is a function of the nodes in the tree. Thus σ reprenets a set of nodes, which consists of vehicle states. We want to find a path σ such that $\sigma(t_0) = x_{init}$, $\sigma(t_f) \in X_{goal}$ and minimize the cost of traversing the path: i.e find σ^* s.t: $c(\sigma^*) = min(c(\sigma), \forall\sigma)$.

The Kinematic Tree (KT) algorithm is shown in Algorithm: 4. The KT builds upon a directed graph $(G = V, E)$ based on the allowable transitions of a vehicle. $V = \{x_1, x_2, ...\}$, is the set of nodes which can be infinite. $E = \{(x_i, x_j) | x_i, x_j \in V\}$ is the set of edges.

The graph is initialized with the start point and start time ($G.init(x_{init})$). All the allowable transitions are added to the tree with time encoded at each node. All the reachable configurations in the tree are called nodes. Each node in the tree encodes an inertial position, velocity, time and a cost function (typically the cost assigned to each node is the cost incurred to reach the node). The edge set E contains the control to get to the node.

For propagation of the tree structure, instead of picking a random configuration from X_{free},

Algorithm 4: The Kinematic Tree Algorithm

```
1  Function :Kinematic_tree(x_init);
2  G.init(x_init);
3  for i=1 to K do
4  |   x_rand_state ← random_state(G);
5  |   Extend(G,x_rand_state)
6  end
7  Return
8  Function :Extend(G,x_rand_state);
9  X_next_states ← Steer(x_rand_state, U, Δt);
10 for all x_state ∈ X_next_states do
11 |   if collision_free_path(x_random_state, x_state) then
12 |   |   G.add_node(x_state);
13 |   |   G.add_edges(x_rand_state, x_state, u))
14 |   end
15 end
16 Return G
```

(as described in Chapter 2 by algorithms like RRT and PRM), a random state (x_{rand_state}) is picked directly from the existing nodes of the Tree (at the first iteration the initial node is selected for expansion). Then the *Steer* function generates the reachable configurations (X_{next_states}) by expansion of the random state using the *motion primitives* (motion primitives are described in the next section). Let U represent the entire set of allowable inputs . Thus for each control $u_k \in U$, applied to the vehicle for a specific amount of time, produces a new state. The set of all possible new states is denoted as X_{next_states}. Now from all the states the ones that are collision free are added to the tree (see Figure 3.1). A cost function is associated with each node. Nodes are selected based on weighted random approach. This process is repeated until the goal is reached.

The propagation of the Kinematic Tree is fundamentally different from that of an RRT algorithm. Rather than selecting a random configuration in X_{free} and finding an input that leads from the existing tree to that random configuration, the KT is propagated by adding the forwards reachable set to a node in the tree. The states are propagated using a set of precomputed states called the motion primitives.

3.1.1 Motion Primitives

The state space and the set of all possible control actions are extremely large. Thus, simplification of model to reduce dimensionality of the state space is required, when a solution has to be computed in real time. Instead of approximate representation of the state the concept of motion primitives was introduced in [66]. A motion primitive defines the trajectory followed by the vehicle by following a specific control input for a specific period of time. By choosing an appropriate set of motion primitives the state space can be significantly reduced without giving up the key performance and maneuverability of the vehicle [67].

A discrete time model is one of the many possible ways of specifying motion primitives. The discrete time can be used to formulate a discrete time state transition equation of the form

$$x_{k+1} = f_d(x_k, u_k) \tag{3.1}$$

in which $x_k = x((k)\Delta t)$, $x_{k+1} = x((k+1)\Delta t)$, and u_k is the action in U that is applied from time $(k)\Delta t$ to time $(k+1)\Delta t$. The function f_d represents an approximation to f.

For computation of motion primitives forward $Euler$ integration is used. The details of constriction of motion primitives for both air and ground vehicles can be found in the next chapters.

In the Kinematic tree algorithm the motion primitives are pre calculated based on vehicle kinematics. At each step environmental factors are added to the the motion primitives.

Thus the state transition equation can be written as:

$$x_{k+1} = f_d(x_k, u_k) + f_e(x,t) \tag{3.2}$$

Here f_e denotes the environmental conditions (dependent both on space and time).

The set of motion primitives thus defines a set of reachable states of a vehicle. A base case with no environmental factors is pre-calculated to save computational effort. Environmental factors are added during the growth of the tree as a time varying feature. This allows for very fast expansion of the tree algorithm.

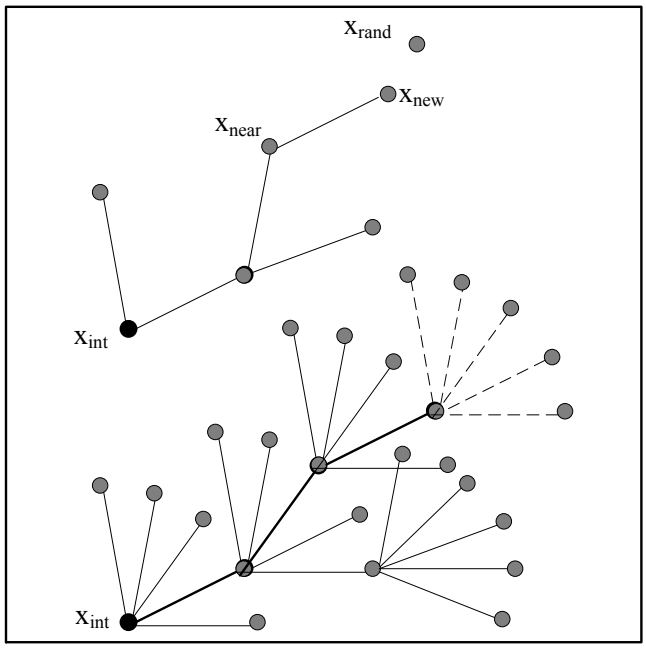


Figure 3.2. Comparison of growth of RC- RRT (upper) and Kinematic Tree (lower)

3.1.2 Resolution Completeness

One of the main complications in using state discretization is that there are three spaces over which sampling occurs: time, the action space and the state space. Often there is trade-off between the three spaces and they can be adjusted depending on requirements. If obtaining optimal solution is important very small discretization should be used. If only feasibility is the requirement, coarser discretization may be used. Resolution complete means if a feasible solution exists at the level of discretization chosen, then it will be found by the planner.

The fundamental difference in which the Kinematic Tree is propagated brings to the obvious question: is there a feasible solution with this method? We will prove that the proposed Kinematic Tree planner is resolution complete: i.e it will find a solution if one exists. In this discussion it is assumed that the discretization used satisfies all the constraints of resolution completeness of RC-RRT. (The area enclosed by X_{goal} should be smaller than the area enclosed between discretized inputs times Δt.)

Lemma 3.1.1. *Every possible realization of an RRT is a subset of all possible realization of an KT. In particular for an identical realization: for all* $i \in N$, $V_i^{RC-RRT} \subset V_i^{KT}$ *and* $E_i^{RC-RRT} \subset$

E_i^{KT}

Lemma 1 implies that the paths discovered by RC-RRT algorithm (discussed in Chapter 2) after each iteration is subset of those discovered by the Kinematic Tree algorithm for an identical realization.

Proof. This will be proved by induction. Figure 3.2 shows the growth of the tree structure of both RRT and Kinematic Tree algorithms.

Let $\mathscr{A}$ represent the set of all possible realization of RC-RRT. Let $\mathscr{A}_z$ represent a particular realization of the ser $\mathscr{A}$.

Let $card(X_{next\_states}) = m$ denote the number of branches in the KT at each iteration, i.e the number of states produced at each iteration is m. At $i = 1$, m branches are added to the Kinematic Tree. Since the path of an RC-RRT comes from the allowable input branches, it must come from the input set of KT. Thus any vertex added to RC-RRT must be one the m branches. Thus at the end of the first iteration for the particular realization $\mathscr{A}_z$, $V_1^{RC-RRT} \subset V_1^{KT}$ and $E_1^{RC-RRT} \subset E_1^{KT}$. Thus the Lemma is true for $i = 1$.

Let the lemma be true for $i = k$ for a particular realizaztion $\mathscr{A}_z$. It will be proven that the lemma is true for $i = k+1$ for that particular realization. Since the lemma is true for $i = k$, $V_k^{RC-RRT} \subset V_k^{KT}$ and $E_k^{RC-RRT} \subset E_k^{KT}$. Now let RC-RRT choose a random point x_{rand}. If the x_{rand_state} chosen by the KT is the same as x_{near} of the RC-RRT, then the lemma is trivially true.

If this is not true, then, there exists $x_{near} \in V_k^{RC-RRT}$. Since $V_k^{RC-RRT} \subset V_k^{KT}$, $x_{near} \in V_k^{KT}$. Now since the lemma is true for $i = 1$, $x_{new} \leftarrow new\_state(x_{near}, u \in U, \Delta t)$ formed by the RC-RRT is a subset of X_{next_states}. Thus for all $i \in N$, $V_i^{RC-RRT} \subset V_i^{KT}$ and $E_i^{RC-RRT} \subset E_i^{KT}$ for the particular realization $\mathscr{A}_z$.

□

Theorem 3.1.2. *If there exists a feasible solution from x_{int} to $x_{goal} \in X_{goal}$ then $lim_{i\to\infty} P(V_i^{RC-RRT} \cap X_{goal} \neq \emptyset) = 1$*

This theorem proves that the RC-RRT is resolution complete and is guaranteed to find a solution if one exists. For proof of theorem 3.1.2 a see [65]

From Lemma 3.1.1 and Theorem 3.1.2 the following theorem is immediate.

Theorem 3.1.3. *The probability that the Kinematic Tree initialized at x_{init} will contain x_{goal} as a vertex approaches 1 as the number of vertices approach infinity*

Hence the Kinematic Tree algorithm is guaranteed to converge to the goal if a solution exists.

3.1.3 Time Complexity

Since m branches are added to the kinematic tree at each iteration, the computational complexity is an important factor determining the applicability of this method. The time complexity analysis of the KT is presented as a comparison with time complexity of RC-RRT as presented in Algorithm 2 in Chapter 2.

3.1.3.1 Time Complexity of KT

For each iteration of the KT, m branches are added to the vertex set and edge set. Thus for N iterations of the KT, mN vertices are added to the memory. The time for N iterations is

$$T(N) = T_{random\_state}(mN) + T_{Steer}(N) + mT_{add}(N) \tag{3.3}$$

where each of T_{random_state}, T_{Steer}, and T_{add} correspond to the *Extend* function in Algorithm 4. T_{random_state} and T_{Steer} are simple operations and can be done in linear time. The *Steer* function corresponds to finding the next states which is a linear function of matrix addition (see Chapter 4 for details) for all the m branches. However the *add* function requires collision checking and has to be done for all the m successors. In the current problem of long-range flight planning only ground collision must be checked, thus only altitude above ground needs to be considered. This can be done in linear time, thus the total time complexity for N iterations is $mO(N)$

The total computational complexity of the kinematic tree is thus

$$T(N) = O(mN) + O(N) + mO(N) \approx O(mN) \tag{3.4}$$

3.1.3.2 Time Complexity Analysis of RRT

The time it takes to add N vertices to the tree (disregarding the initialization), can be calculated as the sum of the time for N iterations of the function *Extend* in Algorithm 2:

$$T(N) = T_{random\_config}(N) + T_{nearest\_neighbor}(N) + T_{new\_state}(N) + T_{add}(N) \tag{3.5}$$

where each T_{random_config}, $T_{nearest\_neighbor}$, T_{new_state}, T_{add} correspond to the *Extend* in Algorithm 2. T_{random_config} and T_{add} can be done in linear time, so the complexity is $O(N)$. The extension time, T_{new_state} is dependent on collision checking but this can be still done in linear time for our application. An extra operation that has to be done in case of RC-RRT is the nearest neighbor.

Each time the nearest vertex has to be found, the distance to all previously added vertices must be calculated:

$$T_{nearest\_neighbor}(N) = \sum(i-1)T_{dist} = \frac{N^2 - N}{2}T_{dist} \tag{3.6}$$

where T_{dist} is the time it takes to calculate the distance to any other vertex. So even though T_{dist} is small compared to T_{next_state}, the complexity is $O(N^2 - N)$. By adding the derived complexity for the $random\_config$, $nearest\_neighbor$, $next\_state$ and add operations respectively, the combined complexity for the brute force nearest neighbor search is:

$$O(N) + O(N^2 - N) + O(N) + O(N) \approx O(N^2) \tag{3.7}$$

It can be seen that the complexity is bounded by $O(N^2 - N)$, which is the complexity for the nearest operation. So when the tree becomes large, a significant part of the computation time is spend on calculating distances to other vertices, and much computing time can be saved if the nearest neighbor search is optimized.

Since number of branches $m << N$, the total number of nodes, the kinematic tree is is computationally more efficient than the RRT algorithm at the additional cost of more memory storage.

Remark 3.1.4. *The kinematic tree algorithm thus provides a solution if one exists. It must be noted that the computation of obstacle avoidance for each of the branches can be computationally expensive (but no more expensive than for any other path planning algorithm), but for the case of aircraft motion planning considered here only height above ground is of concern. Moreover the tree is biased in such a way that the nodes away from obstacles (the ground in this case) are chosen to be expanded. Significant computation time is saved by avoiding sorting of nodes in a near function (as in RRT). The big advantage of using the tree based planner is that time is inherently integrated in the way the tree is built. Also note that the motion primitives can be precomputed to reduce computation time.*

3.2 Optimal Path planning

Algorithm 5: *KinematicTree***Algorithm*

Function :$Kinematic\_tree^*(x_{init})$;
G.init(x_{init});
G.extend $\leftarrow x_{init}$;
G.not_extend $\leftarrow \phi$
for *i=1 to K* **do**
    $x_{rand\_state} \leftarrow choose\_state(G)$;
    Extend(G,x_{rand_state})
end
Return
Function :Extend(G,x_{rand_state});
$X_{next\_states} \leftarrow Steer(x_{rand\_state}, U, \Delta t)$;
for *all* $x_{state} \in X_{next\_states}$ **do**
    if *collision_free_path*($x_{random\_state}, x_{state}$) **then**
        $G.add\_node(x_{state})$;
        $G.add\_edges(x_{rand\_state}, x_{state}, u))\ G.extended(x_{rand\_state}))$
        $G.not\_extended \leftarrow G/G.extended$;
        $G.cost \leftarrow cost(x_{rand\_state}) + g(x_{rand\_state}, x_{state}) + h(x_{state}))$
    end
end
Return G
Function :Choose_State(G,x_{rand_state});
$x\_lowest\_cost \leftarrow find\_min\_cost(G)$
Return x_lowest_cost;

In this section a optimal version of the Kinematic Tree is presented. For flight planning in complicated wind field, one cannot guarantee global optimality of the cost function by the Kinematic Tree algorithm. This is because energy is available form the atmosphere. This makes energy expended by the aircraft while traversing a distance to be negative (i.e. the cost function becomes inadmissible for general wind fields).. However if we choose a cost function which is admissible (e.g the distance cost function in case of a robot navigating a region filled with obstacles to reach the goal), one can show that the Kinematic Tree algorithm can be modified to be an optimal planner with a simple few extra lines of code. This planner can be used for applications like a mobile robot in indoor or outdoor environments or linked manipulators.

Let $c : E \rightarrow R^+$ be the cost assigned to each edge.

$\sigma(x_{init}, x_{goal}) = (x_{init} = x_1, x_2, ...x_k = x_{goal})$ is a path from the initial to the final position. Then the cost along the path is defined as:

$$c(\sigma(x_{init}, x_{goal})) = \sum_{i=1}^{k-1} c(x_i, x_{i+1}) \tag{3.8}$$

We seek to find an algorithm which will find the optimal path σ^* such that $c(\sigma^*) = min(c(\sigma), \forall\sigma)$.

Algorithm 5 presents the optimal version of Kinematic tree algorithm. The main difference between the Kinematic Tree and Kinematic Tree* is: the node that is selected for expansion at each stage in Kinematic Tree*, chooses the cost function which minimizes $\hat{f}(x) = \hat{g}(x) + \hat{h}(x)$, where the total cost f is the sum of cost of navigating to that node g:

$$g(x_i) = \sum_{i=1}^{k-1} c(x_i, x_{i+1}) \tag{3.9}$$

and $\hat{h}$ is the heuristic estimate of the remaining cost to reach the goal. In case of robot navigating a obstacle field environment the cost $\hat{h}$ is the euclidean distance of the current position and the goal. The cost g is the total cost to go to that particular node from x_{init}, essentially g is the total sum of the *Euclidean* distance traversed to reach that node.

3.2.1 Admissibility of Cost Function

A cost function is admissible if the heuristic function describing the estimate of remaining cost to go never "over estimates" the actual cost to go. It means that the optimal estimator $\hat{h} \leq h$ for all the nodes in the search tree.

Consider a pair of nodes such that x_j is a successor of x_i (Figure 3.3). Since the cost function is admissible, we have

$$\hat{h}(x_i) - \hat{h}(x_j) \leq c(x_i, x_j) \tag{3.10}$$

where $c(x_i, x_j)$ is the cost of traversing from x_i to x_j.

Equation 3.10 can also be written as

$$\hat{h}(x_j) \geq \hat{h}(x_i) - c(x_i, x_j) \tag{3.11}$$

This condition states that along any path in a search graph the estimate of optimal (remaining) cost to go cannot decrease by more than the arc cost along the path.

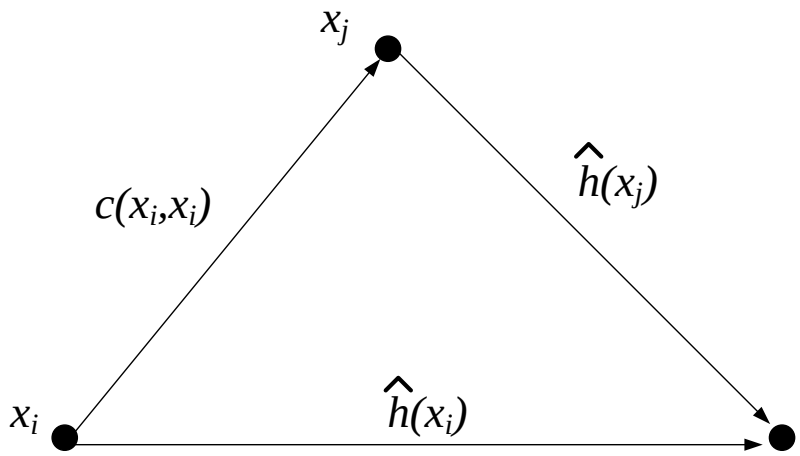


Figure 3.3. Consistency Condition.

Note that for planning in a wind field where there is upward moving air along the path the energy expended along the path is smaller than the "expected " energy expended at zero wind. Thus Equation 3.11 is not valid for planning in a wind field with upward component of wind.

The obvious solution for finding optimal path in a flow field seems, intuitively, to have a heuristic which takes into account upward moving air as a part of heuristic. For example, consider the heuristic to be zero for all the nodes i.e the expected cost is zero for all the nodes. This satisfies the admissibility condition no doubt but results in a breadth first search of the tree being incrementally built. This does not lead to any practical solutions as the solution is exponential in length of the path found (discussed in Section 3.3).

The admissibility condition is usually satisfied for obstacle avoidance problems: the euclidean distance to the goal can be used as an estimate of cost to go and it never over-predicts cost to go. It is also usually satisfied for navigation in rough terrain: friction (or rolling resistance) always results in energy consumption (energy gain is generally not possible).

3.2.2 Optimality of Kinematic Tree* Algorithm

In this section the proof of optimality for the Kinematic Tree* algorithm is presented. It is essential to note that the cost function needs to be admissible to guarantee optimality.

Lemma 3.2.1. *At every step before termination of Kinematic_Tree*, there is always a node* $x^* \in G.not\_extended$ *such that*

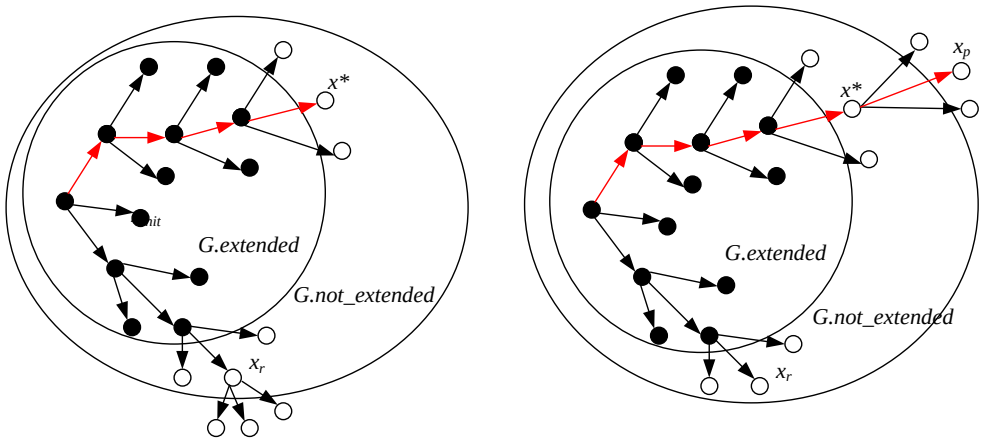


Figure 3.4. Situation when (n+1)th node is selected.

1. x^* *is in the optimal path to goal*

2. *Kinematic_Tree** *has found an optimal path to* x^*

3. $\hat{f}(x^*) \leq f(x_0)$

Proof. This will be proved by induction.

This is true at initiation because the starting point x_{init} is necessarily part of the optimal path. Also, since $\hat{f}(x_{init}) = \hat{h}(x_{init}) \leq f(x_{init})$, this is because the cost function is admissible. Hence, $\hat{f}(x_{init}) \leq f(x_{init})$

Induction Step : Assume the lemma true at step m. To prove that it is true for step $m+1$:

Let x^* be the node on *G.not_extended* which is on an optimal path found by KT* after the m^{th} step (see Figure 3.4. The optimal path is denoted in red). Now, if x^* is not selected for expansion on the $(m+1)^{\text{th}}$ step, x^* has the same property as before , thus proving the induction case for that case. If x^* is selected for expansion all of its successors will be put in *G.not_extended* and at least one of them (x_p), will be on an optimal path, and KT* has found an optimal path to x_p. We let x_p to be the new x^* and we have proved the induction step.

Now for any node x^*, on an optimal path and to which KT* has found an optimal path we have

$\hat{f}(x^*) = \hat{g}(x^*) + \hat{h}(x^*)$ *by definition*
$\leq g(x^*) + h(x^*)$ *because* $\hat{g}(x^*) = g(x^*)$ *and* $\hat{h}(x^*) \leq h(x^*)$
$\leq f(x^*)$ *since* $g(x^*) + h(x^*) = f(x^*)$ *by definition*

$\leq f(x_{init})$ *because* $f(x^*) = f(x_{init})$ *since* x^* *is on an optimal path*

□

Theorem 3.2.2. *Paths Discovered by KT* are optimal up to resolution if the cost function is admissible.*

Proof. To prove this theorem we need to prove that KT* must terminate and terminate by taking the optimal path. Note that KT* is a special case of KT algorithm and we have already proved that KT algorithm is resolution complete. Since the number of branches produced KT is subset of that produced by KT,* KT* is guaranteed to converge. Also since Lemma 3.2.1 is true for each iteration KT* terminates by taking the optimal path to goal. □

3.3 Heuristic Function and Search Efficiency

The Kinematic Tree* algorithm is guaranteed to find an optimal path if the cost function is admissible. Computational complexity or search efficiency is a critical parameter for any algorithm. This section outlines the effect of the heuristic on search efficiency.

Selection of the cost to go heuristic function is crucial in determining the efficiency of the Kinematic Tree* algorithm. The absolute minimum number of node expansion is achieved if we uses a heuristic function identical to the actual cost h. But calculation of such a heuristic is same as solving the original problem.

Let m be the number of branches produced at each step. Let N be the number of total nodes expanded and d is the depth of the tree formed to find the solution. If the length of solution found is d, then in the best case scenario the solution is reached in d steps. Hence the total number of nodes expanded is md, where m is the number of branches laid down at each step. The worst case scenario arises when each leaf of each branch is expanded before getting to the goal.

Thus N is bounded by

$$N \geq md \tag{3.12}$$

$$N \leq m + m^2 + m^3 + ... + m^d \tag{3.13}$$

Hence the worst case complexity of KT* is $O(m^d)$ and the best case complexity is $O(md)$.

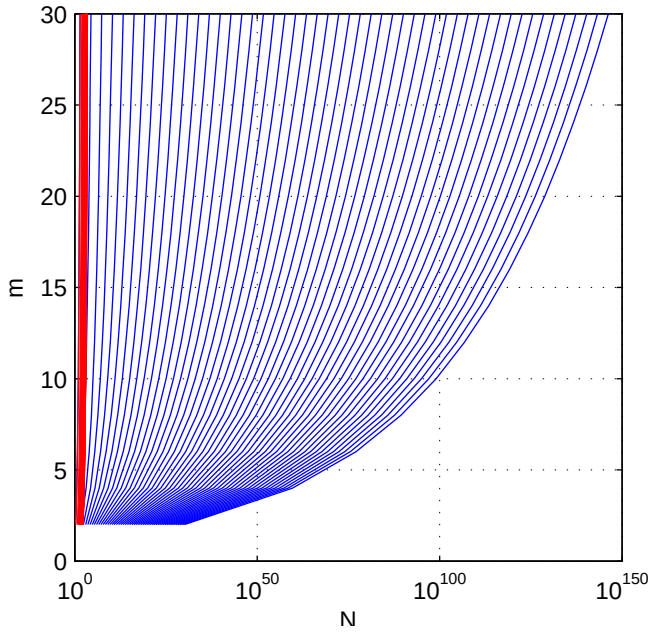


Figure 3.5. Number of nodes expanded vs branches.

This is same as the order of complexity of the KT algorithm assuming that the goal is reached in d steps.

Figure 3.5 shows the number of nodes expanded as a function of branches m for a fixed depth of the tree d. The lines in *red* represent the best and the lines in *blue* represent the worst case scenarios. The time complexity is strongly dependent on the heuristic. As discussed the worst case the number of nodes expanded is exponential in the length of the solution.

Relaxing the optimal requirement can reduce the complexity of KT*. KT* spends a lot of time discriminating between paths whose cost does not vary significantly. Thus there is need to minimize search effort while bounding cost to acceptable ranges.

3.3.1 Adjusting weights of g and h

One of the most common strategies employed to ease computational complexity is to use a weighted evaluation function. For a weighing constant α such that $0 \leq \alpha \leq 1$, a weighted cost function can be represented as:

$$\bar{f}(x) = (1-\alpha) \times g(x) + \alpha \times h(x) \tag{3.14}$$

$\alpha = 0,\ 0.5$ *and* 1 corresponds to breadth first, KT* and depth first search algorithms. It is

easy to show that if h is admissible $\bar{f}$ is admissible in the range $0 \leq \alpha \leq 0.5$ and may lose its admissibility for $0.5 < \alpha \leq 1$. Simulation results show that as $\alpha \to 1$ the cost of solution found deteriorates but the solution converges quickly.

Rather than keeping α constant throughout the search, a more intuitive idea is to dynamically vary the weight according to the depth of the tree. Using Pohl's [68] approach for A^* algorithm a dynamic evaluation function can be used:

$$f(x) = g(x) + h(x) + \alpha \left\{ 1 - \frac{r(x)}{D} \right\} h(x) \tag{3.15}$$

where $r(x)$ is the remaining distance to goal and D is the total distance from start to goal. At shallow depth of the search $r(x) << D$ more weight is placed in the heuristic and at deep levels search becomes admissible. It is easy to show that if $h(x)$ is admissible then the algorithm is α admissible, i.e it finds a path with cost at most $(1+\alpha)C^*$ where C^* is the the optimal cost. This follows from noting that before termination any node x' on $G.not\_extended$ along any optimal path has $g(x') = g^*(x')$ and thus

$$\begin{aligned} f(x') &\leq g^*(x) + h^*(x) + \alpha \left\{ 1 - \frac{r(x)}{D} \right\} h^*(x) && (3.16) \\ &\leq C^* + \alpha h^*(x') && (3.17) \\ &\leq (1+\alpha)C^* && (3.18) \end{aligned}$$

Thus the algorithm cannot terminate with a cost worse than $(1+\alpha)C^*$

3.3.2 Probabilistic Analysis of Complexity of KT*

A significant amount of research has been done in heuristic search in tree based algorithms [69] [68]. The Kinematic Tree, though a sampling based planning algorithm, inherits all the properties of a heuristic search because of its tree structure. Note that the Kinematic Tree algorithm iteratively develops the tree which is searched eventually. This section summarizes some of the results from heuristic based search algorithms which are used to analyze the complexity of Kinematic Tree* algorithm.

When KT* employs a perfectly informed heuristic ($h = h^*$) the number of nodes expanded is minimum and the complexity of operation is linear with the depth of the tree. At the other extreme when no heuristic is available the search becomes exhaustive yielding exponential

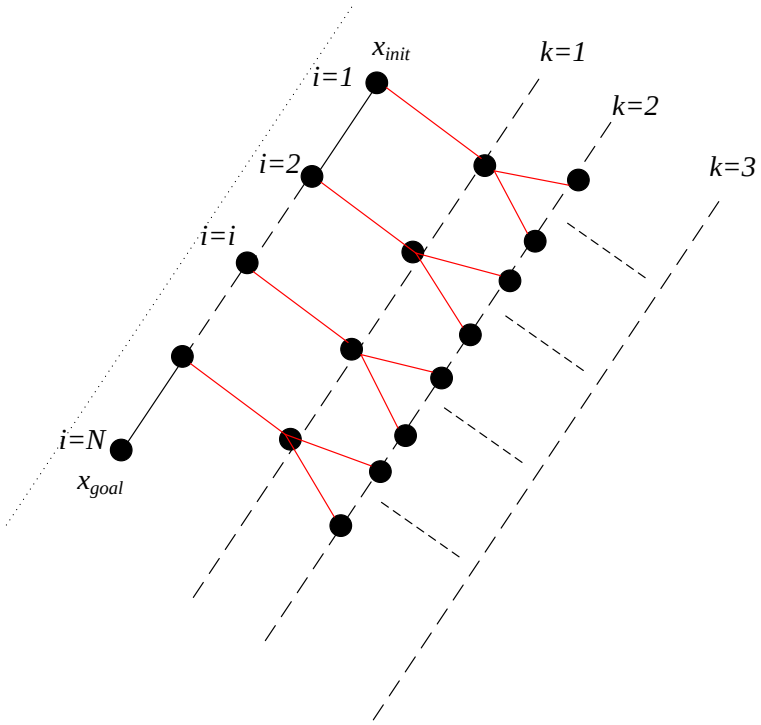


Figure 3.6. Tree model for analyzing complexity of KT*. The goal node is located at a depth of N in the tree. The branches in red denote "of course" branches that does not lead to the solution.

growing complexity. Between this two extremes a relation is sought so that we can bound the complexity based on predictions of heuristic.

3.3.2.1 A General Formula for Mean Complexity of KT*

Following [69] and [68] a probabilistic model for performance analysis of KT* is presented here. Let the heuristic function $h(x)$ be a random variable that is distributed over all the nodes.

The process by which the KT* algorithm expands is analogous to a branching process, where, for any node x not on the solution path the condition for "being expanded" is $f(x) < f^*(x_{init})$ and the condition for termination $f(x) \geq f^*(x_{init})$.

Figure 3.6 shows a simplified branching process of a binary tree. Let x_{init} be the start node and x_{goal} be the goal node located at a depth N in the tree. The "of-course" branches are shown in red as k increases. The "of-course" branches are the branches of the tree that do not lead to a solution.

The branching process model was originally used to study the family survival problem[70],

where branching is analogous to reproduction. At the first iteration the 0^{th} generation has only one member(x_{init}). This member gives rise to m members (m is the number of branches developed at each step). Now all of the m members may not survive. (Some die due to infeasibility from collision check, traversibility etc.) Then the surviving members give rise to random number of offspring of second generation and this goes on.

Let the i^{th} member of generation k give birth to a random number $X_{i,k}$ of "fertile descendants" (surviving nodes which can expand further in the tree structure) which are members of generation $k+1$. Thus the size s_{k+1} of the $(k+1)^{th}$ generation is given by

$$s_{k+1} = X_{1,k} + ... + X_{i,k} + ... + X_{s_k,k} \tag{3.19}$$

Now the number of offspring produced in each generation is independent of the generation. Thus:

$$E(s_{k+1}) = E(s_k)E(X_{i,k}) \tag{3.20}$$

Let p_k denote the probability of survival of an off spring at generation k. Thus $X_{i,k}$ is a binomial random variable with mean

$$E(X_{i,k}) = mp_k \tag{3.21}$$

and thus

$$E(s_{k+1}) = E(s_k)mp_k \tag{3.22}$$

By induction over $k = 0, 1, 2....$ we obtain

$$E(s_k) = m^d p_k p_{k-1} .... p_0 \tag{3.23}$$

Thus is true for all of course branches which satisfies the admissibility condition.

Thus the total expected number of nodes expanded is

$$E(Z) = N + (m-1) \sum_{i=1}^{N} \sum_{d=0}^{N} m^d \prod_{k=1}^{d} p_{i,k} \tag{3.24}$$

Equation 3.24 shows that the expected number of nodes expanded depends on $p_{i,k}$, which is the probability of expanding the nodes and depends on the heuristic chosen.

3.3.2.2 Bounds on Complexity

Equation 3.24 gives a general formula for the mean complexity of KT*. It has been shown in [71] that if the relative error of the heuristic, i.e $[h^*(n)-h(n)]/h^*(n)$ remains constant, then the search complexity of a tree algorithm is exponential. But if the absolute error $[h^*(n)-h(n)]$ remains constant the search complexity is linear [71].

Following [68], it can be shown that for any error distribution on $[h^*(n)-h(n)]$, if KT_2^* is stochastically more informed than KT_1^* , then KT_2^* is stochastically more efficient than KT_1^*, i.e

$$Z_2 \leq Z_1 \; if \; h_1(n) \leq h_2(n) \tag{3.25}$$

Thus a better informed heuristic is computationally more efficient. Thus better estimates of cost-to-go will significantly reduce the computational cost of the KT* algorithm.

One of the most important results proved by Pearl [68] for tree algorithms is:

$$E(Z) = \begin{cases} O(c^N) & \text{if } P(h = h^*) < 1 - \frac{1}{m} \\ O(N) & \text{if } P(h = h^*) \geq 1 - \frac{1}{m} \end{cases} \tag{3.26}$$

This result implies the only time the complexity of KT* reduces to polynomial is when h coincides with h^* with high probability. Thus if the difference is distributed over all nodes the complexity is exponential in N.

Thus if we can bound the total error we can guarantee linear search but is the error in heuristic grows with nodes the search becomes exponential.

Between these two extremes, it can be shown that, a necessary and sufficient condition for maintaining polynomial search by KT*, is that the growth of the error be guided by a heuristic with logarithmic precision: $\phi(n) << O(log(n))$ [68].

For a complete survey of results presented in this section see [72]

Thus the complexity of Kinematic Tree* grows exponentially if the error in the heuristic is not bounded. Thus for practical applications careful consideration of the heuristic has to be taken into account. To achieve polynomial time complexity the heuristic has to be bounded or have to be sacrificed for non optimal solutions.

3.4 Summary

This chapter has presented the theoretical foundations of the proposed Kinematic Tree algorithm. The Kinematic Tree algorithm is based on sampling based motion planning. Instead of sampling the free space, the Kinematic Tree algorithm samples the input space of the controller.

The algorithm is proved to be resolution complete: i.e it is guaranteed to find a solution if the resolution parameter is tuned fine enough. Time complexity analysis of the algorithm is presented and is compared with the computational complexity of the RRT algorithm. The algorithm is shown to be more efficient than the RRT algorithm but requires more memory.

The algorithm is extended to be optimal, if the cost function to be minimized is admissible. Admissibility of the cost function implies that the expected cost-to-go is never over estimated. This cannot be guaranteed for aircraft path planning in complex environments. However for planning for ground robots to avoid obstacles this is true. The Kinematic Tree* algorithm is the optimal version of the Kinematic Tree algorithm.

Since the Kinematic Tree algorithm is a tree search it inherits many properties of heuristic based tree search algorithm. This provides a guide to choose admissible heuristics with polynomial time complexity.

Chapter 4 shows simulation results for Kinematic Tree algorithm, where the KT is able to find feasible paths in presence of time varying complex wind field. Chapter 5 shows the simulation results for KT* algorithm for ground robots.

Chapter 4

Kinematic Tree For Flight Planning

This chapter describes the simulation results of an UAV flight in *mountain wave* data. The simulations serves two main purposes: first to demonstrate that Kinematic Tree algorithm can effectively handle 3-D time varying winds to plan feasible paths in complex environments; second, to find energy efficient paths over a complicated wind field.

Mountain wave is an atmospheric phenomena which is observed in mountainous regions such as the Appalachian mountains over central Pennsylvania. This chapter will introduce mountain waves and meteorological tools which enabled us to get wind information *a priori*.

The previous chapter has defined the generic Kinematic Tree algorithm. The motion primitives of a Kinematic Tree planner are based on the kinematics of the robot. In this chapter how motion primitives are built based on the kinematics of an aircraft will be described. How nodes are chosen for expansion is elaborated. This chapter will show how the Kinematic Tree can be used in practice.

Two different simulation results are presented to demonstrate the efficacy of the Kinematic Tree planner. First the Kinematic Tree is evaluated to find feasible paths towards the goal of a glider using only energy from the atmosphere. Next, simulation results for a Nimbus III DM glider is presented to gain maximum altitude. Note gliders do not have engines to propel them and thus the feasible path can be found only by utilizing energy available from wind.

Section 4.1 describes the mountain wave phenomenon as observed over Central Pennsylvania and how meteorological modeling provides wind data *a priori*. Section 4.2 describes the kinematics of a glider. Section 4.3 describes how the Kinematic Tree can be constructed from the kinematics of the glider discussed in Section 4.2. Section 4.4.1 shows the results obtained from the simulation results. Section 4.4.2 discusses another set of simulation results

Figure 4.1. A wave over the Bald Eagle Valley of central Pennsylvania. Source: Wikipedia

for a different mission using the same Kinematic Tree framework.

4.1 Mountain Wave

In meteorology, *lee waves* are atmospheric standing waves produced by long oscillations of the atmosphere and found mostly in lee of mountains. The most common form of lee waves is mountain waves, which are atmospheric internal gravity waves.

Mountain waves are periodic changes of atmospheric pressure, temperature and orthometric height in a current of air caused by vertical displacement of air. The vertical displacement of air can be caused by different factors, for example by orographic lift: when the wind blows over a mountain or mountain range, it produces orographic lift. They can also be caused by the surface wind blowing over an escarpment or plateau, or even by upper winds deflected over a thermal updraft or cloud street.

The vertical motion forces periodic changes in speed and direction of the air within this air current. They always occur in groups on the lee side of the terrain that triggers them. Usually a turbulent vortex, with its axis of rotation parallel to the mountain range, is generated around the first trough; this is called a rotor. The strongest lee waves are produced when the lapse rate shows a stable layer above the obstruction, with an unstable layer above and below.

Lee waves provide a possibility for gliders to gain altitude or fly long distances when soar-

ing. World record wave flight performances for speed, distance or altitude have been made in the lee of the Sierra Nevada, Alps, Patagonic Andes, and Southern Alps mountain ranges.

The conditions favoring strong lee waves suitable for soaring are:

- A gradual increase in wind speed with altitude
- Wind direction within 30 degrees of perpendicular to the mountain ridgeline
- Strong low-altitude winds in a stable atmosphere
- Ridgetop winds of at least 20 knots

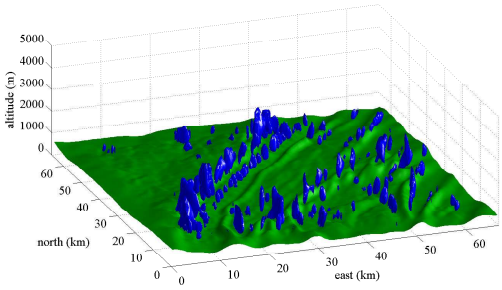
Figure 4.2 shows a wave window over the Bald Eagle Valley of central Pennsylvania as seen from a glider looking north. The wind flow is from upper left to lower right. The Allegheny Front is under the left edge of the window, the rising air is at the right edge, and the distance between them is 3 to 4 km. Central Pennsylvania holds promise to autonomously fly gliders to exploit mountain wave.

4.1.1 Meteorological Modeling

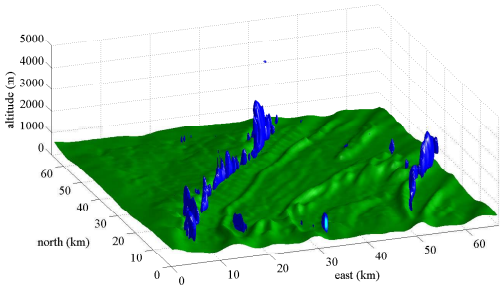
The meteorological modeling support for this research was provided by the Numerical Weather Prediction (NWP) group of the Department of Meteorology at Penn State University. The core of the realtime modeling system is the Advanced Research Weather Research and Forecasting (WRF) model (WRF-ARW) [73]. The WRF modeling system is a state-of-the-science community-supported numerical weather prediction (NWP) and atmospheric simulation system .

Real wind fields can be extremely complex, exhibiting significant temporal as well as spatial variation. Trajectory generation to exploit atmospheric energy becomes correspondingly complex. A high-fidelity simulation of a wind field is used as a test case for the problem. Wind field data was generated using WRF-ARW (Weather Research and Forecasting-Advanced Research WRF) version 2.2, and simulates the development of ridge lift and mountain wave over central Pennsylvania.

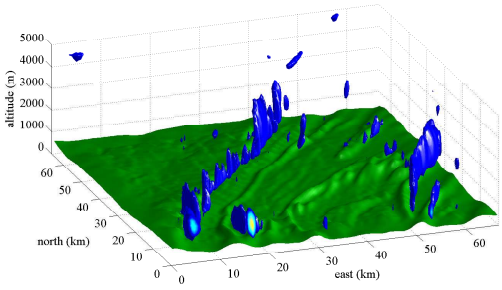
The wind field used as the unsteady example for the simulations in this chapter is shown in Figure 4.2. The wind field gives east, north, up components of wind at 0.44km grid spacing horizontally. The vertical resolution is descending in nature with more density near the surface and gradually decreasing with altitude. There are 42 vertical terrain following layers. The



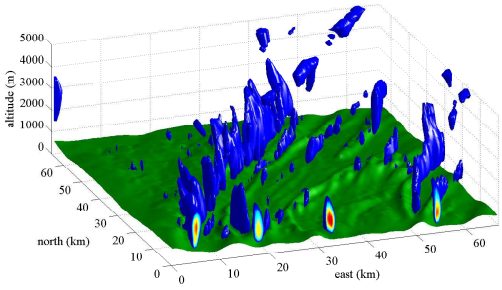
(a) t=0000h UTC



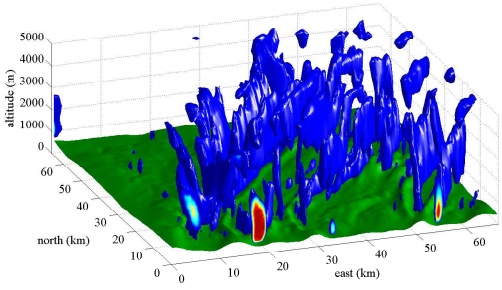
(b) t=02000h UTC



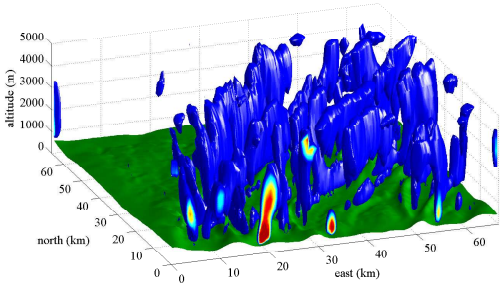
(c) t=0400h UTC



(d) t=0600h UTC



(e) t=0800h UTC



(f) t=1100h UTC

Figure 4.2. Visualization of wind field data for a complex time varying wind field.

WRF vertical layers are based on dry hydrostatic pressure. Wind is provided in intervals of 15 minutes starting at 0000 UTC on October 7,2007 and ending at 1200 UTC on October 7, 2007. This wind field is a computational study of an actual event: the development of mountain wave over central Pennsylvania [74].

Figure 4.2 shows a visualization of this wind field plotting regions where energy can be harvested (i.e. where the vertical component of the wind speed is greater than the minimum sink rate of the aircraft). Blue isosurfaces bound energy harvesting regions, with subfigures (a) through (f) showing the time evolution of the wind field. Note the significant spatial as well as temporal variation of the wind field, leading to a particularly challenging planning problem. Clearly a "good" path planning algorithm will find trajectories that fly through these regions while avoiding regions of downwards moving air.

4.2 Kinematics of Soaring Flight

The previous section has defined the complex wind field which will be used to show the efficacy of Kinematic Tree algorithm. The focus of this section is to define the kinematics of the glider which will be used to construct the tree based planning framework.

It is assumed that an on-board controller, aboard the vehicle, is capable of several control modes, including constant airspeed flight, constant heading flight and constant bank angle (i.e. turning) flight. It is also assumed that the response to step changes in commands is fast compared with the duration of a particular command. Hence a point mass model is sufficient to describe vehicle motion for planning purposes.

Vehicle kinematics are given by

$$\dot{x} = v_a \cos\gamma \cos\psi + w_x \tag{4.1}$$

$$\dot{y} = v_a \cos\gamma \sin\psi + w_y \tag{4.2}$$

$$\dot{z} = v_a \sin\gamma + w_z \tag{4.3}$$

$$\dot{\psi} = \eta \tag{4.4}$$

where v_a is the air speed, γ the glide path angle ψ is the heading and and η is the rate of change of heading. $\mathbf{w} = [w_x\ w_y\ w_z]^T$ is the 3D wind vector.

Consider the forces acting on a glider which is in a steady bank angle of ϕ and flight path

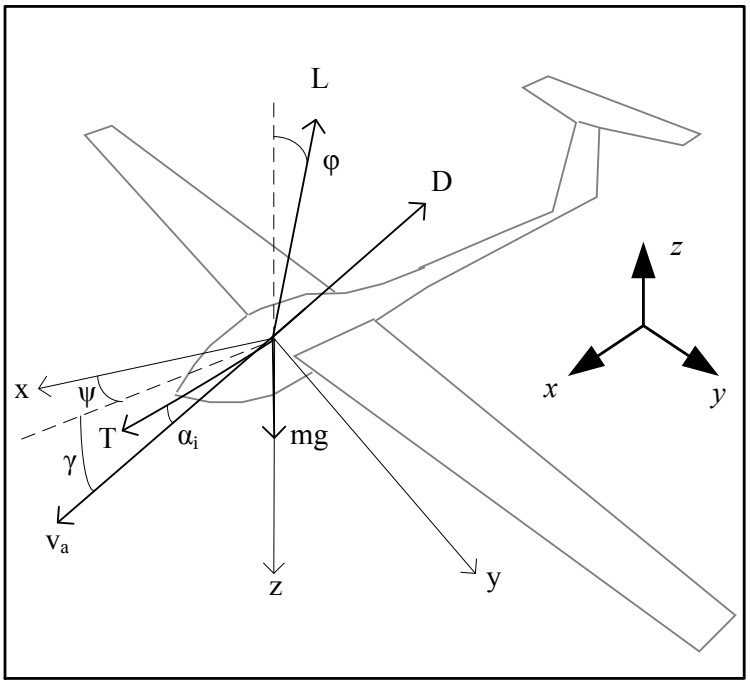


Figure 4.3. Point mass model.

angle γ (Figure 4.3). The glide path angle γ is a function of airspeed v_a and throttle setting T and can be obtained for steady flight. Equating the forces in parallel and perpendicular to the the flight path,

$$mg\sin\gamma = D - T\cos\alpha_i \tag{4.5}$$

$$mg\cos\gamma = L\cos\phi + T\sin\alpha_i \tag{4.6}$$

where α_i is the incidence angle between thrust vector and the flight path and m is the mass of the vehicle. Assuming small flight path angle γ and $\alpha_i = 0$ (i.e, thrust is aligned to the flight path angle, so that thrust has negligible contribution to force perpendicular to flight path),

$$mg\gamma = D - T \tag{4.7}$$

$$mg = L\cos\phi \tag{4.8}$$

Using lift coefficient defined as $L = \frac{1}{2}\rho v^2 S C_L$,

$$C_L = \frac{2mg}{\rho v^2 S\cos\phi} \tag{4.9}$$

The drag coefficient can be expressed as a polynomial function of lift coefficient:

$$C_D = \sum_{i=0}^{n} a_i C_L^i \tag{4.10}$$

and with $D = \frac{1}{2}\rho v^2 S C_D$ the flight path angle is thus

$$\gamma = \frac{\rho v^2 S}{2mg}\sum_{i=0}^{n} a_i C_L^i - \frac{T}{mg} \tag{4.11}$$

To find the turn rate consider the horizontal components of the forces perpendicular to the flight path.

$$L\sin\phi = m v_a \dot{\psi} \tag{4.12}$$

Dividing Equation 4.8 by Equation 4.12 we have,

$$\tan\phi = \frac{v_a\dot{\psi}}{g} \tag{4.13}$$

thus,

$$\dot{\psi} = \frac{g\tan\phi}{v_a} \tag{4.14}$$

Thus the general equations of motion become

$$\dot{x} = v_a\cos\gamma\cos\psi + w_x \tag{4.15}$$

$$\dot{y} = v_a\cos\gamma\sin\psi + w_y \tag{4.16}$$

$$\dot{z} = v_a\sin\gamma + w_z \tag{4.17}$$

$$\dot{\psi} = \frac{g\tan\phi}{v_a} \tag{4.18}$$

Thus the flight path is completely specified by thrust (T), velocity (v_a), heading angle (ψ) and bank angle (ϕ) and the wind vector. Typical commercially available autopilots such as the Piccolo [75] are able to accept airspeed and heading commands as well as airspeed and bank angle commands. Note that one would not simultaneously command a bank angle and a heading angle if coordinated flight is to be maintained: in general the heading angle command loop would use bank angle to obtain a turn rate to maintain the desired heading.

4.3 The Kinematic Tree Construction for Path Planning

The Kinematic Tree for flight planning is described in detail in this section. The tree is initialized at the vehicle start position and time. From this start configuration the tree is expanded by computing a set of reachable configurations based on vehicle kinematics. At the start position one of the allowable motion primitives is zero velocity, so that the position remains constant but time varies. This encodes the possibility of delaying launch until a more favorable time.

The set of configurations that are reachable after some time interval Δt defines nodes in the tree. Each node encodes inertial position, time, heading, airspeed, a cost for that node and the distance from the node to the goal.

$$n_i = [x_i \; y_i \; z_i \; t_i \; \psi_i \; v_{a,i} \; C_i \; r_{goal,i}] \tag{4.19}$$

The tree is expanded incrementally by selecting a node and computing the set of configurations reachable from that node. To save computation time the motion primitives are pre-defined and computed based on a set of allowable inputs.

4.3.1 Motion Primitives

The set of motion primitives used to build the tree is pre-computed to reduce the computational time during incremental build of the tree.

A particular input u $\in$ U (where U is the set of allowable inputs) is

$$\mathrm{u}_{ijkl} = [T_i \; v_j \; \Delta\psi_k \; \phi_l] \tag{4.20}$$

where each component is chosen from a discrete set of allowable inputs:

$$T_i \in [T_1\ T_2\ T_3\ ...\ T_I] \tag{4.21}$$

$$v_{a,j} \in [v_{a,1}\ v_{a,2}\ v_{a,3}\ ...\ v_{a,J}] \tag{4.22}$$

$$\Delta\psi_k \in [\Delta\psi_1\ \Delta\psi_2\ \Delta\psi_3\ ...\ \Delta\psi_K] \tag{4.23}$$

$$\phi_l \in [0\ \phi_L] \tag{4.24}$$

where $\Delta\psi$ represents a change in heading at the beginning of the segment followed by straight line flight for the remainder of the segment. The bank angle command is either zero or a user-specified bank angle(ϕ_L) that results in a reasonable steady turn. To ensure "reasonableness" of the reachable configurations certain restrictions are are placed on allowable combinations:

bank angle command is either zero or a user-specified bank angle that results in a reasonable steady turn.

$$0 \leq T_i \leq T_{max} \tag{4.25}$$

$$v_{a,min} \leq v_{a,j} \leq v_{a,max} \tag{4.26}$$

$$\text{if } \phi_l \neq 0 \text{ then } \Delta\psi_k = 0 \text{ and } v_{a,j} = v_{a,min} \tag{4.27}$$

Here $v_{a,min}$ is chosen to be the minimum power flight condition (equivalent to minimum sink speed for a glider.) This is chosen to maximize altitude gain while circling flight mode is observed. Note that $\phi \neq 0$ means that the aircraft is in steady turn, which allows the aircraft to keep latitude and longitude nearly constant while altitude changes. This is the typical circling flight to gain altitude when thermals are observed by gliders. Further note that if non-zero bank angle is selected, then the initial heading change is zero (so that the flight path is either a turn followed by straight line flight or a steady turn).

Given a choice of input $\mathbf{u} \in \mathbf{U}$ and a time Δt the motion primitives are computed using numerical integration.

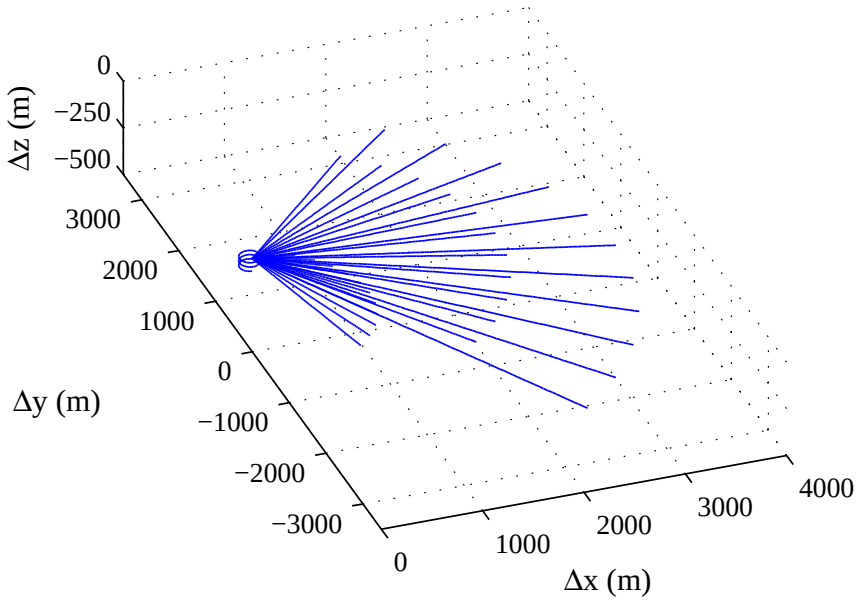


Figure 4.4. Motion Primitives in zero Wind.

$$\Delta x^b_{i,j,k,l} = \begin{pmatrix} x^b_{ijkl} \\ y^b_{ijkl} \\ z^b_{ijkl} \\ \psi^b_{ijkl} \end{pmatrix} = \int_0^{\Delta t} \begin{pmatrix} v_a \cos\gamma \cos\psi \\ v_a \cos\gamma \sin\psi \\ v_a \sin\gamma \\ \frac{g\tan\phi}{v_a} \end{pmatrix} dt \tag{4.28}$$

Note that wind speed **w** is not included in this set of precomputed branches: it is included when a particular node in the tree is expanded. The set of branches X^b consists of all possible changes in position and altitude given the set of possible inputs U, and is formed by concatenating the vectors of branches:

$$\Delta X^b = [\Delta \mathrm{x}^b_{1111} \ldots \Delta \mathrm{x}^b_{IJKL}] \tag{4.29}$$

The motion primitives for the SB-XC glider used in simulations are computed using the following sets of allowable inputs and $\Delta t = 120$ seconds .

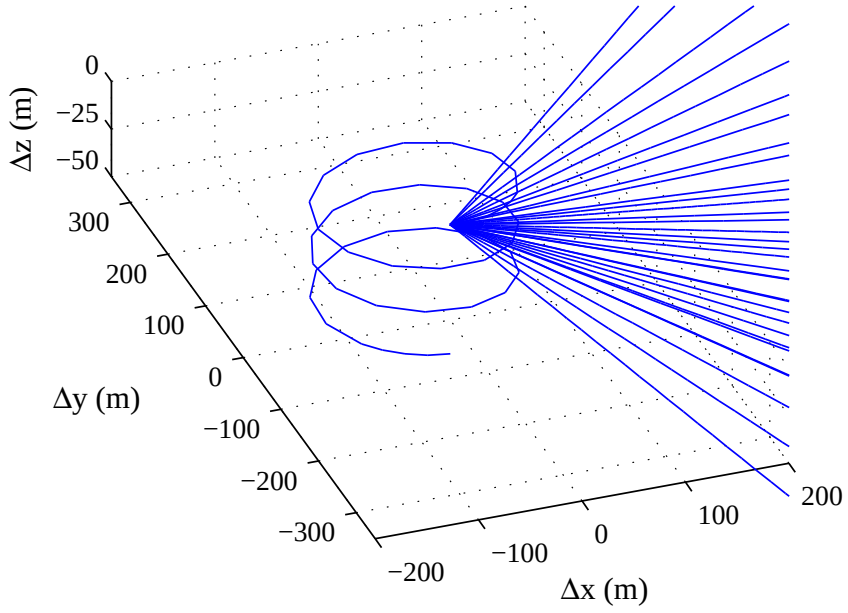


Figure 4.5. Spiral Motion Primitives in zero Wind.

Figure 4.4 shows a set of motion primitives computed for a small soaring capable UAV (based on the RnR Products SB-XC: parameters are given in Appendix B). The segment time Δt is 120 seconds and the set of allowable inputs is

$$T = 0 \tag{4.30}$$

$$V_a = [15\ 20\ 25\ 30\ 35] \tag{4.31}$$

$$\Delta\psi = [-50^\circ\ -40^\circ\ ...\ 40^\circ\ 50^\circ] \tag{4.32}$$

$$\phi = [0^\circ\ 30^\circ] \tag{4.33}$$

Note the small spiral motion primitive given by the steady bank angle motion (Figure 4.5). In steady state bank, the air velocity of the glider is the best sink rate airspeed. Thus the loss in altitude for this motion primitive is smallest compared to other motion primitives. This spiral motion essentially allows altitude change with no change in x and y co-ordinates.

4.3.2 A note on Resolution

Both the time interval and the chosen speeds and heading changes affect resolution of the resulting tree. There is a clear trade between resolution and the time required to find a solution (high resolution generally leads to a longer time to find a solution) and the memory required to store the tree. The practical upper bound on resolution is thus ultimately limited by available computing hardware.

Note also that the resolution of the wind field can have an effect on desired tree resolution. In the simulations presented here the wind field is available on a grid of 444m x 444m. The resolution on z varies with altitude with around 100 m near the ground to 1000m at 25 km altitude. The average flight speed of the SB-XC is about 20 m/s, thus a time step of 120 seconds means that roughly 2400m is traversed in each segment (and approximately 100m altitude would be lost in still air over each segment). The resolution was so chosen that it would capture the variation of wind along the path but would keep the number of nodes expanded to be tractable in terms of memory utilization.

Although it was possible to compute flight plans at higher spatial resolution , this did not appreciably have an effect on the path computed. A tighter discretization on time was used for simulation with Nimbus III-DM glider as average speed of the glider is much faster compared to the SB-XC.

4.3.3 Node Selection and Expansion

Nodes are selected on the basis of survival of the fittest. A cost function C_i dictates which nodes are selected for expansion. The cost function C_i assigned to each node is energy altitude h_E divided by the distance to goal. where,

$$h_E = h + \frac{v_a^2}{2g} \tag{4.34}$$

and

$$C_i = \frac{h_E}{r_{goal}} \tag{4.35}$$

The cost function ensures that the nodes chosen minimize the distance to the goal and

maximize the total energy (either in terms of altitude gain or velocity gain).

Selecting the node with the lowest cost leads to a depth-first search: if there are few "dead ends" or local minima then a solution will be found quickly. But as with all depth first search the problem of local minima can affect the time of computation of a feasible solution.

To remove the problem of being stuck in a local minima a weighted random approach may be taken. Instead of choosing the node with $max\{C_i\}$ a weight $w_i = C_i^2$ is added to each node. The probability of choosing a particular node from the group is proportional to its weight. The weighted random approach biases the search in a direction likely to lead to a feasible solution but has the potential to explore the entire space. The value of the exponent on the weight w_i serves to stretch the range values and can thus be used to increase the likelihood of choosing a higher weighted node by increasing the difference between the smallest and the largest node.

The selected node with position x_i is expanded using the pre-computed branches and wind speed to define a set of candidate nodes.

$$X_{i,new} = \mathrm{x}_i\mathbf{1} + \mathrm{T}_i\Delta X^b + \mathrm{w}_i\Delta t\mathbf{1} + \begin{pmatrix} 0 \\ 0 \\ \Delta z_i \end{pmatrix} \tag{4.36}$$

where $\mathbf{1}$ is a $1 \times IJKL$ array of ones, w_i is the wind vector computed at x_i , and

$$\mathrm{T}_i = \begin{pmatrix} cos\psi_i & sin\psi_i & 0 \\ -sin\psi_i & cos\psi_i & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{4.37}$$

is the transformation which rotates the set of precomputed branches to the local frame defined by the heading ψ_i. Finally $\Delta z_i = [\Delta z_{i,1}\ \Delta z_{i,2}\ \ldots\ \Delta z_{i,IJKL}]$ with

$$\Delta z_{i,j} = \frac{v_{a,i}^2 - v_{a,j}^2}{2g} \tag{4.38}$$

The term Δz_m accounts for the change in altitude which occurs with a change in speed, assuming that total energy is constant during the transition. This is not reflected in the kinematic

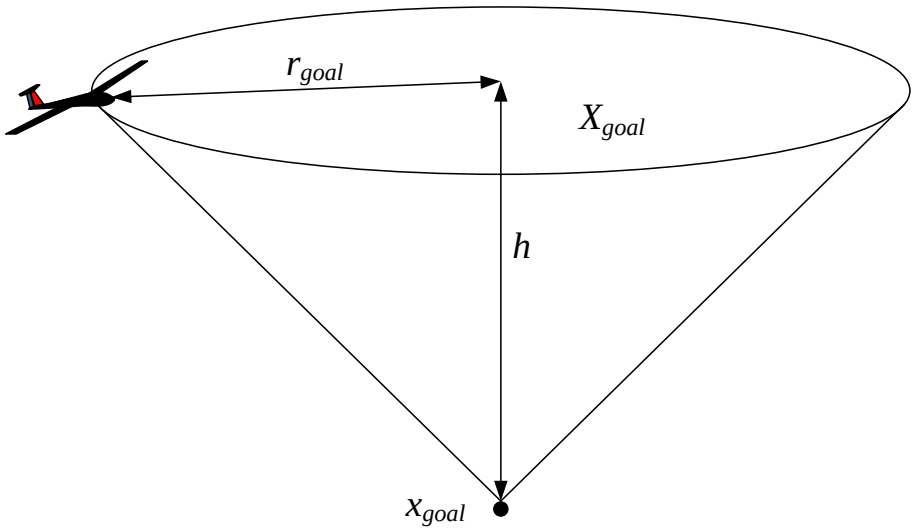


Figure 4.6. Endgame Region.

model of the aircraft and must therefore be accounted for separately.

The wind vector w_i is computed at x_i and is assumed to be constant over the time Δt required to fly the branch.

4.3.4 End Game

The tree is terminated and successful path is reported once the tree finds a node within the gliding distance of the goal. This can be expressed as $r = (L/D)h$, where L/D is the glide ratio and h is the altitude above the goal (see Figure 4.6). Maximum glide ratio occurs at particular airspeed which depends on the wind speed. The end game region (X_{goal}) is defined as

$$X_{goal} = \left\{ x : \frac{r_{goal}}{\delta t} \leq V_{nom} \;\&\; \frac{r_{goal}}{\Delta h} \leq \frac{L}{D} \mid_{nom} \right\} \tag{4.39}$$

One can choose V_{nom} and $\frac{L}{D} \mid_{nom}$ to specify the size of the end game region in terms of nominal vehicle performance so that reaching of goal is ensured in all wind conditions.

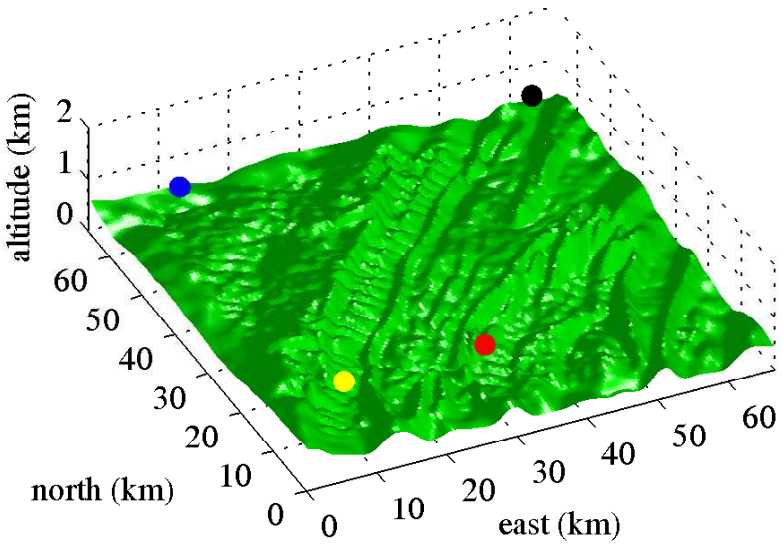


Figure 4.7. Scenario For simulation. The goal is shown by the black dot and start regions are shown by blue,yellow and red dots.

4.4 Simulation Results

The previous section has outlined the Kinematic Tree for a glider. This section will present the the simulation results of the algorithm in the mountain wave data. Two sets of the simulation results are presented. The first simulation finds feasible paths from a known starting location and the problem is to find a path to the known destination using energy that can be derived from wind. The second simulation tries to find the maximum altitude that can be gained by a commercial glider by using only updrafts from the mountain wave data.

4.4.1 Feasible Paths to the Goal using Wind energy

Here a flight to a distant goal is considered from different starting positions (Figure 4.7). The starting locations are *blue* [10 60 1.5] km, *yellow* [10 10 1] km and *red* [30 10 1] km. The goal location is a distant [60 60 1]km given by the *black* dot. The average ground elevation is approximately 500 m. Thus given the starting altitude of 1000m above sea level (500 m above average ground elevation), the maximum gliding distance is only 12 km (at best L/D). Clearly to reach the goal wind energy has to be utilized.

The starting locations are chosen widely separated so that the effect of wave on successful

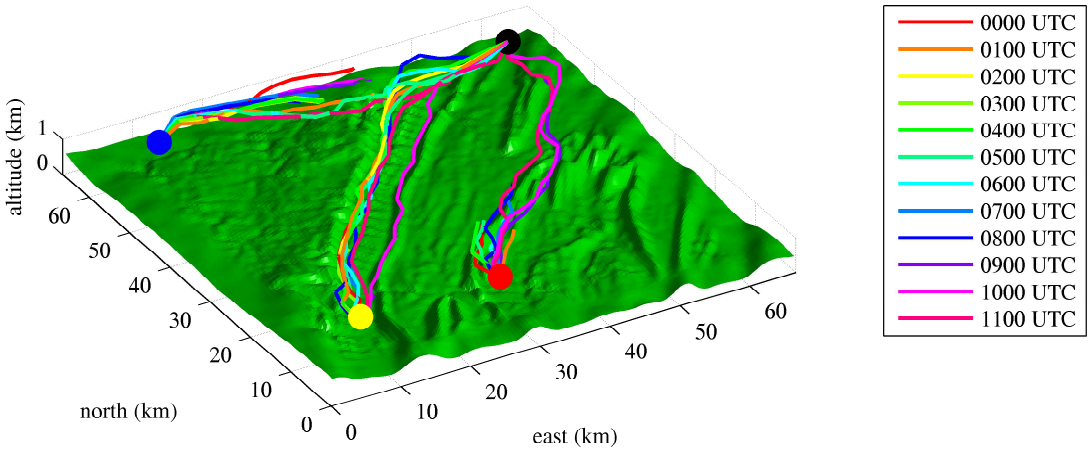


Figure 4.8. Trajectories starting at Different Times of Day.

completion of the paths can be monitored. A vehicle representative of the RnR Products SB-XC is used here: parameters are given in Table B.1. Simulations were carried out on 2.6GHz dual core Intel processor.

Table 4.1. Parameters for SB-XC glider.

variable	value	description
m	10 kg	mass
S	1 m^2	wing area
$f(C_L)$	$0.1723C_L^4 - 0.3161C_L^3 + 0.2397C_L^2$ $-0.0624C_L + 0.0194$	
$v_{a,min}$	12 m/s	
$v_{a,max}$	35 m/s	
$L/D\|_{max}$	25	best glide ratio

Figure 4.8 shows all the paths starting from different starting locations at different starting times. Reachability of goal varies considerably with starting location and start time. While almost all the paths starting from start location *yellow* reach the goal, paths starting only at specific times make it to the goal from start locations *blue* and *red*. Because the winds are both temporally and spatially varying the feasibility of gliding (i.e. unpowered flight) from the various start positions to the goal changes with time. Note that the starting altitude of the *blue* starting point is higher than the other two. Simulation results have shown that no paths reach the goal for *blue* starting points with altitude of 1 km.

Table 4.2 tabulates the earliest start times from the different locations that successfully reach the goal. Paths starting at the point *blue* reaches the goal only for starting times of 0600

Table 4.2. Simulation Results for Different Start Points.

Start Point	**Earliest departure time for successful arrival**	**Time of flight**
blue	0600 UTC	36 mins
red	0900 UTC	42 mins
yellow	0100 UTC	32 mins

Table 4.3. Time of travel and Minimum Distance to goal for trajectories starting at different times of the day for the *yellow* start point

Start Time	**Time of Travel**	**Closest Distance to goal**	**Status**
UTC	(minutes)	(km)	
0000	12	52.030	failed
0100	42	0.0	X_{goal} reached
0200	40	0.0	X_{goal} reached
0300	12	51.220	failed
0400	42	0.0	X_{goal} reached
0500	48	0.0	X_{goal}reached
0600	50	0.0	X_{goal} reached
0700	12	54.898	failed
0800	48	0.0	X_{goal} reached
0900	4	61.123	failed
1000	36	0.0	X_{goal} reached
1100	42	0.0	X_{goal}reached

UTC , 0700 UTC and 1100 UTC . Paths starting at any other times fail to rach the goal from *blue* starting point.

Paths starting at *red* reaches the goal only after the wave has fully developed. All the paths starting after 0900 UTC reach the goal. Thus to reach the goal from the *red* starting point flights only after 0900UTC has to be considered.

Paths starting at *yellow* are analyzed in detail next.

Table 4.4 shows the comparison of the flights forced to start at different times of day (i.e. the null transition was disallowed) for the start point *yellow*. As seen from the results, trajectories starting only at specific time of the day actually reach the goal. Among the paths that reach the goal, the one starting at 0100 UTC reach the goal fastest. Thus to make this flight the optimal start time of travel would be 0100 UTC, while the time of travel is shortest for trajectories starting at 1000 UTC. The effect of wind on the paths computed is clearly visible(Figure 4.8).

Table 4.4. Time of travel and Minimum Distance to goal for trajectories starting at different times of the day for the *yellow* start point

Start Time	**Time of Travel**	**Closest Distance to goal**	**Status**
UTC	(minutes)	(km)	
0000	12	52.030	failed
0100	42	0.0	X_{goal} reached
0200	40	0.0	X_{goal} reached
0300	12	51.220	failed
0400	42	0.0	X_{goal} reached
0500	48	0.0	X_{goal}reached
0600	50	0.0	X_{goal} reached
0700	12	54.898	failed
0800	48	0.0	X_{goal} reached
0900	4	61.123	failed
1000	36	0.0	X_{goal} reached
1100	42	0.0	X_{goal}reached

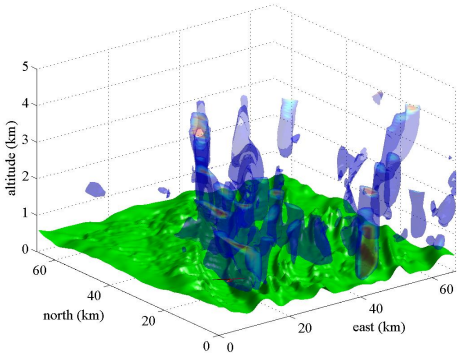
The paths tend to follow the ridges showing evidence of ridge lift along the ridges. Some of the paths, the ones starting at 0300, 0700 and 0900 UTC, end very quickly because there is no vertical air motion of sufficient strength near the start point at those times..

Note that there are no feasible straight-line gliding flight paths to the goal from any of the starting positions at any time. To reach the goal, the aircraft must exploit energy available in the environment.

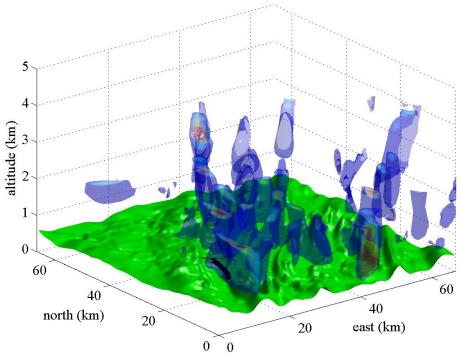
Figure 4.9 shows the time evolution of the trajectory of the flight starting at at 1100 UTC. Recall that the blue isosurface regions are the regions where the glider can gain energy. Clearly the planner was able to utilize these regions of upward moving air.

Figure 4.10 shows the parameters of the trajectory of the flight starting at at 1100 UTC. Clearly the planner was able to find the upward component of wind along the path. The planner is able to hold altitude while reaching the goal as can be seen by the altitude along the path. Not that at the end of the flight the aircraft has to cross a region of downward moving air to reach the destination. Thus the Kinematic Tree successfully finds a path even if it has to cross a “bad” region to get to the goal.

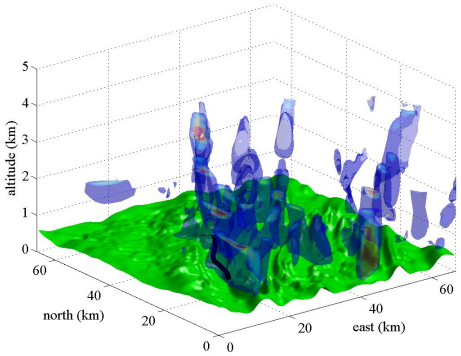
Note that the wind updates after every 15 minutes and thus the the change in environment is updated only in 15 minutes intervals. This represents approximately 14 km traveled between wind field time updates. Note that having winds available at tighter time discretization would



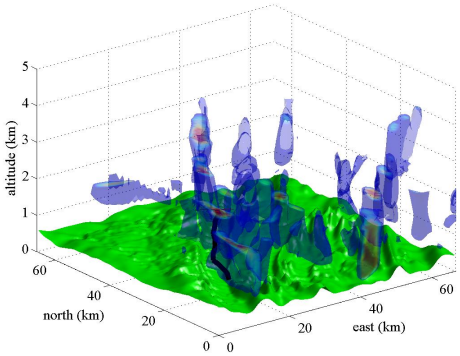
(a) t=1100h UTC



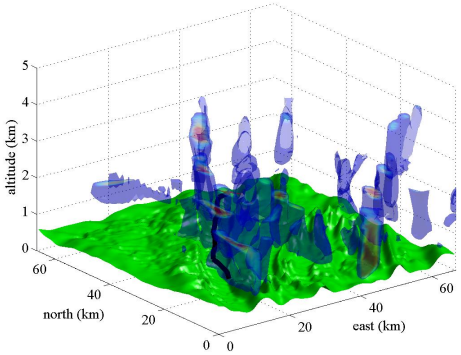
(b) t=1108h UTC



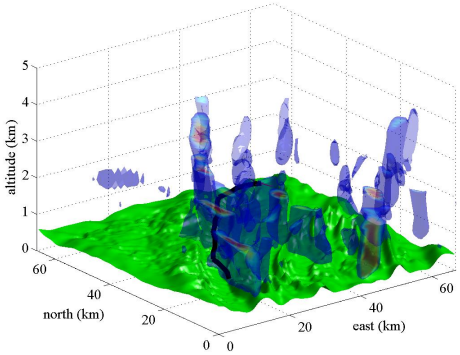
(c) t=1116h UTC



(d) t=1125h UTC



(e) t=1133h UTC



(f) t=1142h UTC

Figure 4.9. Flight starting at 1100 UTC for *yellow* starting position.

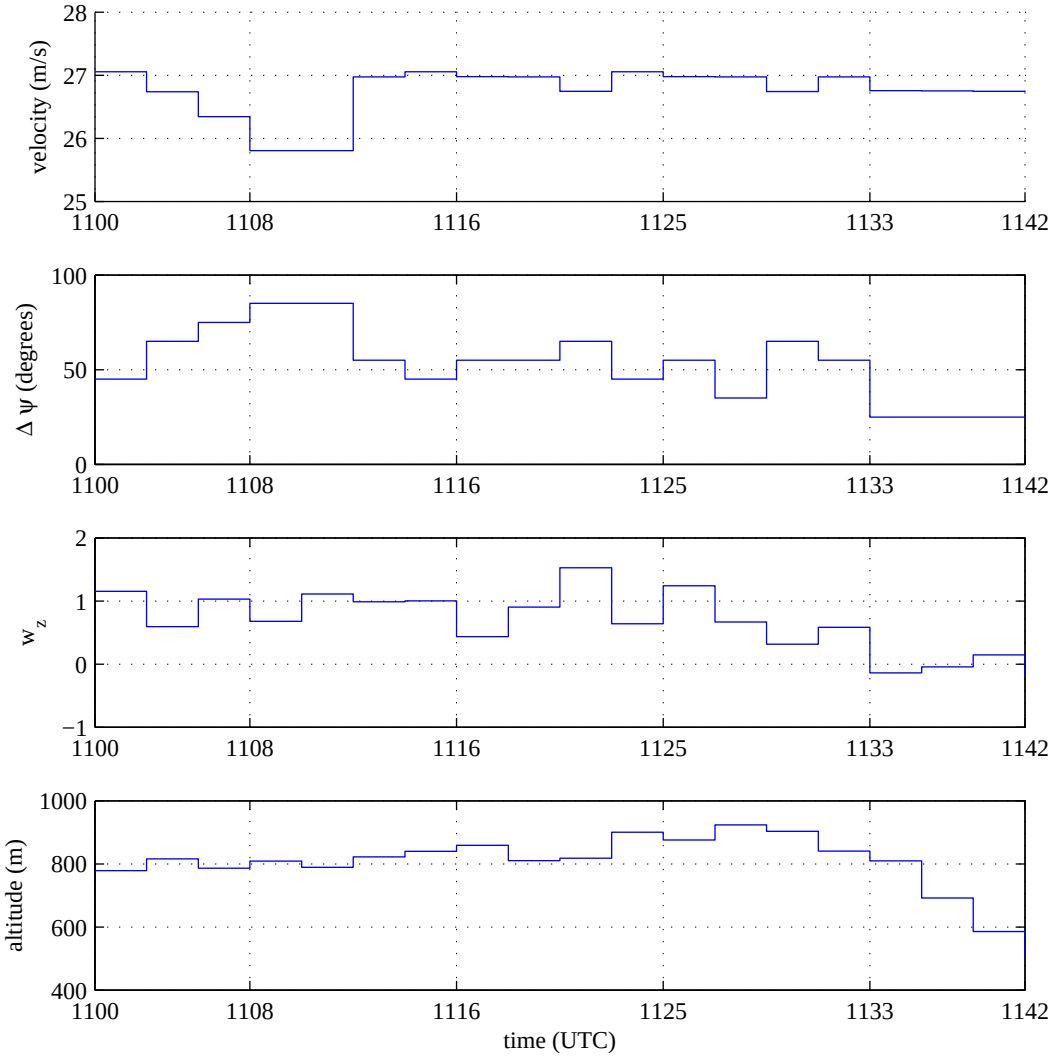


Figure 4.10. Flight starting at 1100 UTC for *yellow* starting position: time history of velocity, heading, vertical component of wind and altitude.

improve performance.

Thus the Kinematic Tree algorithm successfully finds paths to the goal using only energy from the atmosphere. The algorithm also finds possible starting time when such a path is feasible and fastest time in which the goal can be reached. The algorithm truly explores and exploits the wind filed to find paths to the goal.

Figure 4.11. The Nimbus III-DT. Source: http : //francoise.fischer.free.fr/n3/d_kexx.jpg

4.4.2 Maximum Altitude Gain by Nimbus III DM

Gaining altitude by glider pilots using wind wave is practiced around the world both for scientific missions as well as a sport. World record for altitude, speed and distance have been made the lee of the Sierra Nevada, Alps and Andes mountain ranges. Current world record for highest altitude by a glider is held by Steve Fossett and Einar Enevoldson for climbing to 50,699 feet (15,453 m) on August 29, 2006 over El Calafate, Argentina in their modified DG-505.

The Perlan Project is a current research project to fly a glider to an altitude of 90,000 feet. The project was conceived by Einar Enevoldson, a former NASA test pilot, who sought to demonstrate the feasibility of riding stratospheric standing mountain waves.

In this section we test the feasibility of altitude gain by a robotic glider using wind wave. The Kinematic Tree framework will be tested to gain altitude by extracting energy from the wind.

The Nimbus III DM (Figure 4.11 shows a variant of the Nimbus III variation named the Nimbus III-DT) is a high performance two-seat glider. This glider is widely used by glider pilots for various competitions. We use the parameters provided by manual to simulate a flight by the Nimbus glider.

Table 4.5 tabulates the properties of the Nimbus glider. The goal of this simulation is maximum altitude attainable by the glider using mountain wave data.

The motion primitives for the Nimbus III-DM glider used in simulations are computed

Table 4.5. Parameters for Nimbus III DM.

variable	value	description
m	820 kg	mass
S	16.85 m^2	wing area
p	24.6 m	wing span
$v_{a,min}$	30 m/s	
$v_{a,max}$	65 m/s	
$L/D\|_{max}$	57	best glide ratio

using the following sets of allowable inputs and $\Delta t = 60$ seconds. Note the tighter resolution on time for this simulation. This is because the Nimbus III-DM glider is much faster the SB-XC glider used for the previous simulations. Thus to effectively use the wind field a tight resolution in time is required.

$$T = 0 \tag{4.40}$$

$$V_a = [30\ 35\ 40\ 45\ 50\ 55\ 60\ 65] \tag{4.41}$$

$$\Delta\psi = [-50^\circ\ -40^\circ\ ...\ 40^\circ\ 50^\circ] \tag{4.42}$$

$$\phi = [0^\circ\ 30^\circ] \tag{4.43}$$

Cost function assigned to each node at the end of each iteration is the altitude attained. Thus $C_i = h_i$ for the simulation.As before, nodes with the highest cost are selected for expansion. The same wind data as shown in Figure 4.2 is used in the simulations.

Figure 4.13 shows the simulation results. The algorithm successfully finds lift sources from the mountain wave data to gain altitude. Starting form an altitude of 2000m above sea level, the glider gains an altitude of about 5000 m during the entire span of simulation. The simulations only consider an un-powered glider, thus all the altitude gain can be attributed to energy gained by exploiting the vertical component of wind.

Figure 4.13 shows the time evolution of the flight trajectory. The starting time of the simulation was 0600 UTC and the end time was 1200 UTC. Figure 4.13 shows the evolution of mountain wave through out the day over Central Pennsylvania and that the Kinematic Tree algorithm successfully exploits the mountain wave to gain altitude. The trajectory of the glider reveals the strong mountain wave in the lee ward side of the Allegheny mountain ranges which is exploited by Kinematic Tree Algorithm.

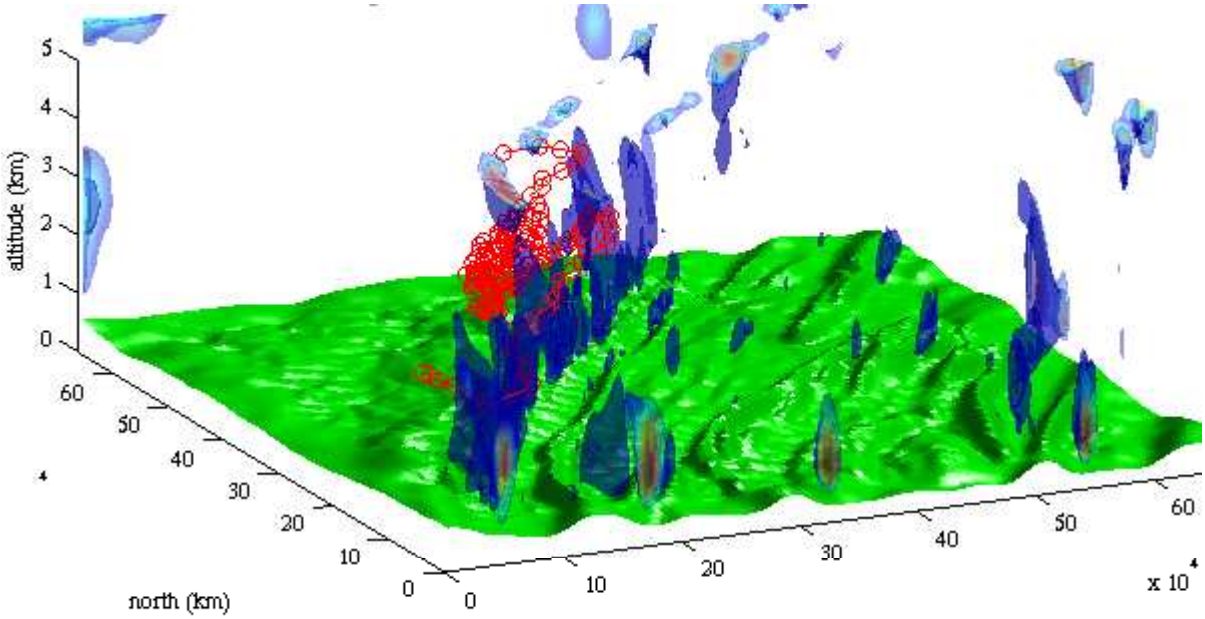


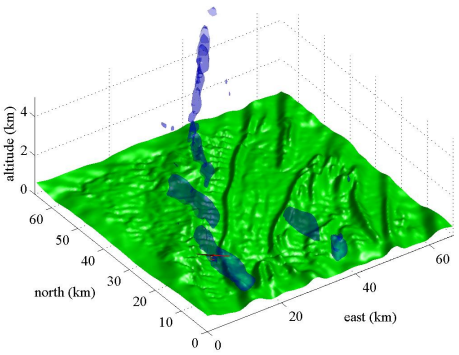
Figure 4.12. Altitude Gain by Nimbus III-DM using only wave

Figure 4.14 shows the time history of the gliders velocity, heading, altitude and the vertical component of the wind along the path. Note the velocity of the Nimbus III-DM glider is around 30 m/s, the best L/D speed of the glider, which results in maximum altitude gain. The plot of vertical component of wind shows that the planner was successfully able to find regions where there is upwards moving air. There is steady gain in altitude over the time span of the simulations which shows the the Kinematic Tree algorithm was successful in finding regions where free lift is available from the environment.

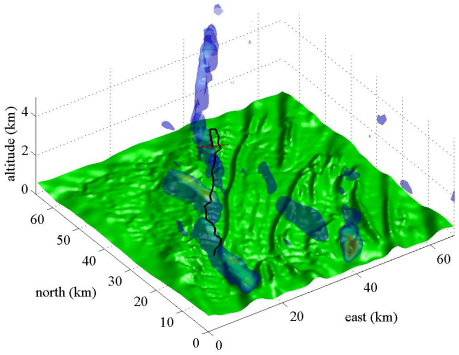
Thus Kinematic Tree algorithm is successfully able to find suitable regions in the atmosphere from which energy can be harvested to gain altitude. The simulation results have shown that the planner is able to handle variations of wind over large period of time and can be used for different cost functions suitable for different applications

4.5 Summary

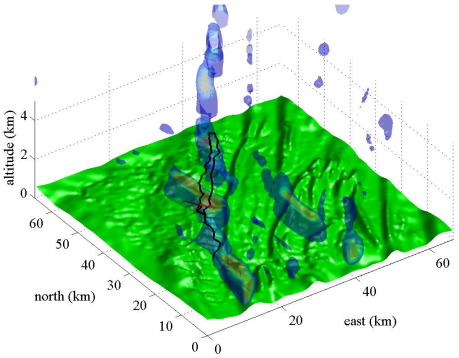
This chapter has presented the simulation results of the Kinematic Tree algorithm proposed in the previous chapter. The simulation results shows that the Kinematic Tree algorithm can effectively handle 3-D time varying winds to plan feasible paths in real wind data. The construction of the Kinematic Tree from the kinematics of an aircraft is shown in details. Initialization,



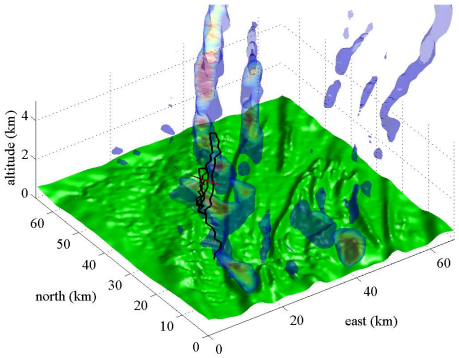
(a) t=0600h UTC



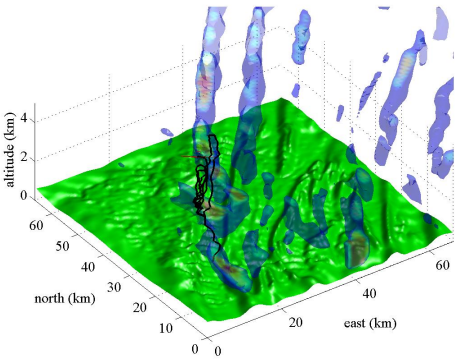
(b) t=0700h UTC



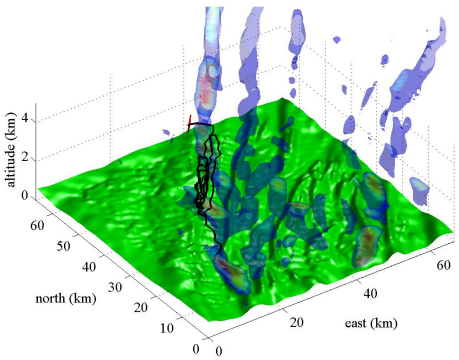
(c) t=0800h UTC



(d) t=0900h UTC



(e) t=1000h UTC



(f) t=1100h UTC

Figure 4.13. Time Evolution of Maximum Altitude Gain Flight.

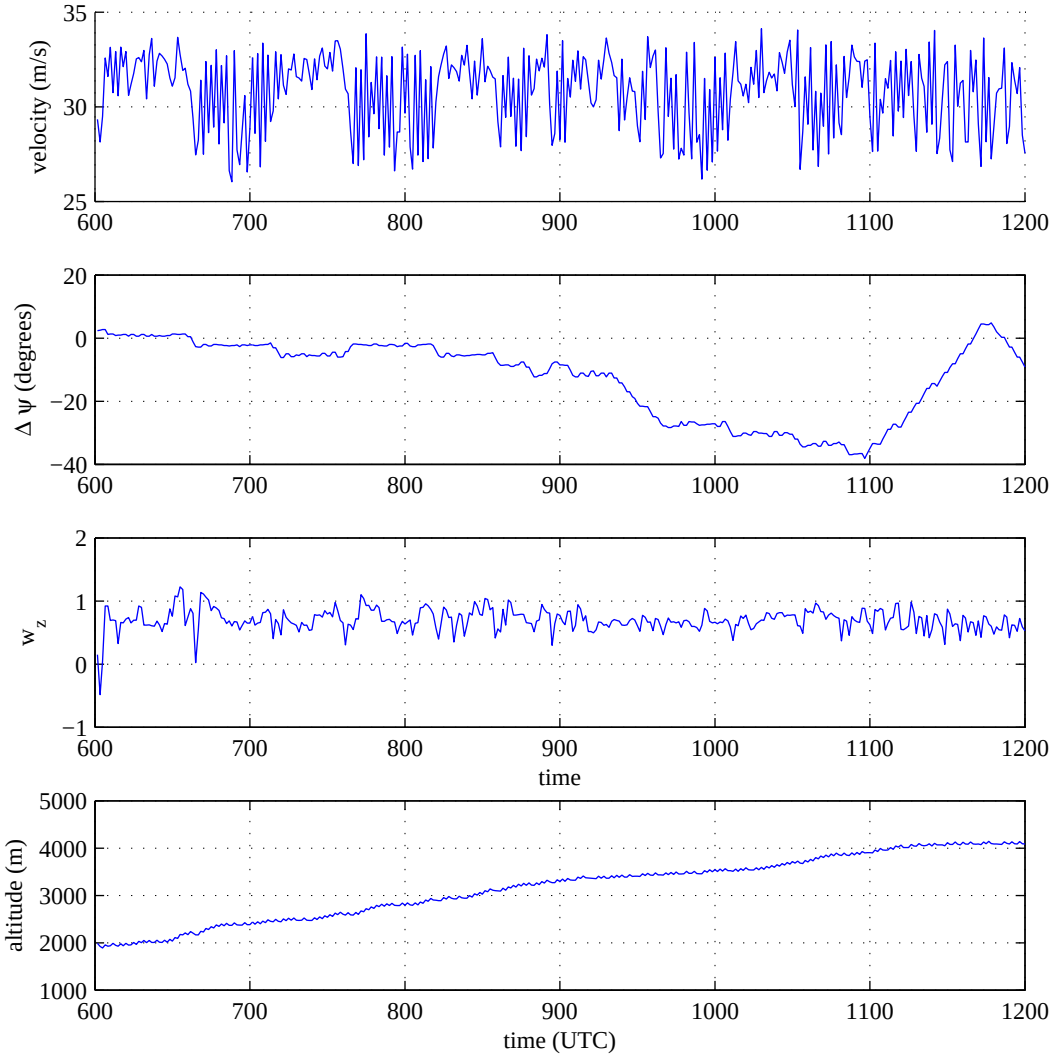


Figure 4.14. Altitude Gain by Nimbus III-DM using only wave: time history of velocity, heading, vertical component of wind and altitude.

growth, selection of nodes and termination of the tree has been described in details. How time varying wind is incorporated in the growth of the tree is shown in details.

Complexity of the Kinematic Tree algorithm is a function of number of branches at each stage of iteration. Number of branches of the Kinematic Tree is a function of resolution of the inputs. Feasibility of paths may depend on resolution. In practical planners there is always a trade-off between computation and completeness.. In practical planners there is always a

trade-off between computation and completeness. Resolution of paths ultimately is decided by available computing power available. The rationale behind the resolution used in this planning problem is discussed which resulted in successful computation of paths.

Two sets of simulation results are presented. The first set of simulations plans feasible paths to reach a destination which can be achieved by utilizing energy from the atmosphere. Simulation results find appropriate starting time and paths to reach a destination using energy from wind. The second set of simulations tests a different cost function with much larger planning horizon. The planner was shown to gain altitude over period of six hours. The simulation results verify tha the Kinematic Tree algorithm can successfully utilize time varying winds to find "energy aware" paths in real environments.

Chapter 5

Kinematic Tree for Optimal Path Planning

Chapter 3 introduced the optimal version of the Kinematic Tree algorithm named the Kinematic Tree* (KT*). This chapter describes the simulation results for the Kinematic Tree* algorithm for ground robots. This chapter will validate the the KT* algorithm through simulation results. The trade between optimality and computational complexity as described in Chapter 3 will be evaluated through simulation results.

The KT* algorithm returns optimal paths when the cost function is admissible. For path planning for ground robots, the distance covered by the vehicle is a typical example of an admissible cost. The euclidean distance cost function obeys the triangle inequality and is a consistent cost function. Hence the Kinematic Tree* algorithm should be able to find optimal paths to the goal.

Obstacle avoidance and path planning in one of the fundamental problems in robotics. Obstacle avoidance has many potential applications like the autonomous future cars, autonomous rovers like MSL and MER. Many algorithms exits for path planning in obstacle field. Particularly algorithms like RRT has been very successful in dealing with path planning in an obstacle field. But optimal path planning in sampling based motion planning is an emerging field of study. The KT* algorithm is a sampling based method which finds optimal paths to the goal.

A challenging problem in ground robotics is path planning in complex terrain. In this case, the environment may include obstacles (that cannot be traversed), slopes (that may be traversed at the cost of higher required power), and changes in ground properties (for example, pavement, grass, or mud) that also change the power required by affecting rolling friction. Recall that an

admissible cost function is one that does not over-predict the cost to go: thus as long as rolling friction does not change drastically then the energy required to traverse terrain is an admissible cost function. In this case KT* can be used to find minimum energy paths.

Section 5.1 describes the obstacle avoidance problem. Section 5.1.2 and Section 5.1.3 describes effects of heuristic on computational complexity of the KT* algorithm. Section 5.2 describes the path planning on mountainous terrain and finally Section 5.3 presents the concluding remarks.

5.1 Obstacle Avoidance

This section presents the obstacle avoidance scenario and the goal is to find optimal paths to reach a destination. It is assumed that the entire description of the world is known beforehand (i.e the location of obstacles are known *a priori*) and it will be shown that the Kinematic Tree* algorithm will find optimal paths to the goal.

Vehicle speed is kept constant and we want to find paths that reflect minimum distance traversed by the vehicle. The distance cost function is admissible since distance traveled is always non-negative, additive and maintains the triangle inequality. Note that, instead of distance traveled, time taken by the the vehicle is an equivalent cost function, if the speed is kept constant.

5.1.1 The Algorithm

The Kinematic Tree* algorithm is again repeated here for completeness. It is assumed that both the start x_{init} and the goal points x_{goal} belong the free space X_{free} and a feasible path exists between the start and the goal.

Algorithm 6 shows the Kinematic Tree* algorithm. The tree is initialized by the start node x_{init} and a set of branches are precomputed based on the kinematic model of a ground vehicle. All the allowable transitions, which are collision free, are added to the node. The cost of transition is the distance covered at constant speed. At each step of iteration the node which is picked for expansion minimizes the total distance traveled up to that node and the expected remaining distance left. The expected remaining distance is the simple euclidean distance between the end point of the node and the goal. The nodes which end up in collision are assigned infinity cost function and are never picked for expansion.

Algorithm 6: The optimal Kinematic Tree algorithm.

Function :KinematicTree*(x_{init});
G.init(x_{init});
G.extended $\leftarrow x_{init}$;
G.not_extended $\leftarrow \phi$
for *i=1 to K* **do**
    $x_{rand\_state} \leftarrow choose\_state(G)$;
    Extend(G,x_{rand_state})
end
Return
Function :Extend(G,x_{rand_state});
$X_{next\_states} \leftarrow Steer(x_{rand\_state}, U, \Delta t)$;
for *all* $x_{state} \in X_{next\_states}$ **do**
    if *collision_free_path*($x_{random\_state}, x_{state}$) **then**
        $G.add\_node(x_{state})$;
        $G.add\_edges(x_{rand\_state}, x_{state}, u))\ G.extended(x_{rand\_state}))$
        $G.not\_extended \leftarrow G/G.extended$;
        $G.cost \leftarrow cost(x_{rand\_state}) + g(x_{rand\_state}, x_{state}) + h(x_{state}))$
    end
end
Return G
Function :Choose_State(G,x_{rand_state});
$x\_lowest\_cost \leftarrow find\_min\_cost(G)$
Return x_lowest_cost;

5.1.1.1 The Unicycle Car Model

A discretized unicycle model of car is considered. The kinematic equations can be written as:

$$\dot{x} = v_g \cos\psi \tag{5.1}$$

$$\dot{y} = v_g \sin\psi \tag{5.2}$$

where v_g is the ground speed and ψ is the heading. It is assumed that an on-board controller is able to steer the vehicle along this path.

A particular input u $\in$ U (where U is the set of allowable inputs) is

$$\mathrm{u}_k = \begin{bmatrix} v_g & \Delta\psi_k \end{bmatrix} \tag{5.3}$$

As in the case of aircraft motion planning $\Delta\psi$ represents a change in heading at the beginning of the segment.

The motion primitives for the ground vehicle used in simulations are computed using the following sets of allowable inputs and $\Delta t = 120$ seconds.

$$v_g = 25m/s \tag{5.4}$$

$$\Delta\psi = [-50^\circ \ -40^\circ \ ... \ 40^\circ \ 50^\circ] \tag{5.5}$$

All the nodes at the end of each iteration is assigned a cost function $f(n) = g(n) + h(n)$ where:

$$g(n) = \sum_{i=0}^{n} (\Delta s_i) \tag{5.6}$$

$$h(n) = \Delta s_{i,rem} \tag{5.7}$$

where, Δs_i represent the length of each segment of the motion primitives and $\Delta s_{i,rem}$ represent the expected remain distance.

Nodes are selected based on the heuristic cost function $\hat{f}(n) = \hat{g}(n) + \hat{h}(n)$ where $\hat{h}(n)$ and $\hat{g}(n)$ represents the estimate of the values of $h(n)$ and $g(n)$ respectively.

Figure 5.1 shows the scenario for planning. The start point is located at (14 km, 5 km) and the destination goal is at (60 km,60 km). There are four obstacles located at $(10km \geq x \leq 18km, 8km \geq y \leq 18km)$, $(22km \geq x \leq 26km, 8km \geq y \leq 12km)$, $(22km \geq x \leq 26km, 22km \geq y \leq 26km)$, $(31km \geq x \leq 44km, 31km \geq y \leq 44km)$.

Figure 5.1 shows a representative run of the Kinematic Tree* Algorithm for static obstacles. As shown in Figure 5.1 the Kinematic Tree* successfully finds near optimal paths to the goal. Because of discretization the path is not optimal, but tighter resolution in motion primitives will lead to paths that are nearer to optimal at the cost of longer convergence time and greater memory requirements. The effect of discretization on the found paths is evident. A tighter resolution on the discretization will lead to high resolution optimal paths at the cost of higher convergence time.

The rate of convergence of the algorithm is directly related to the discretization of the KT*. One approach is to modify the algorithm to find low resolution path initially and then once a path has been discovered it can be refined further. In this way a faster near optimal path can be

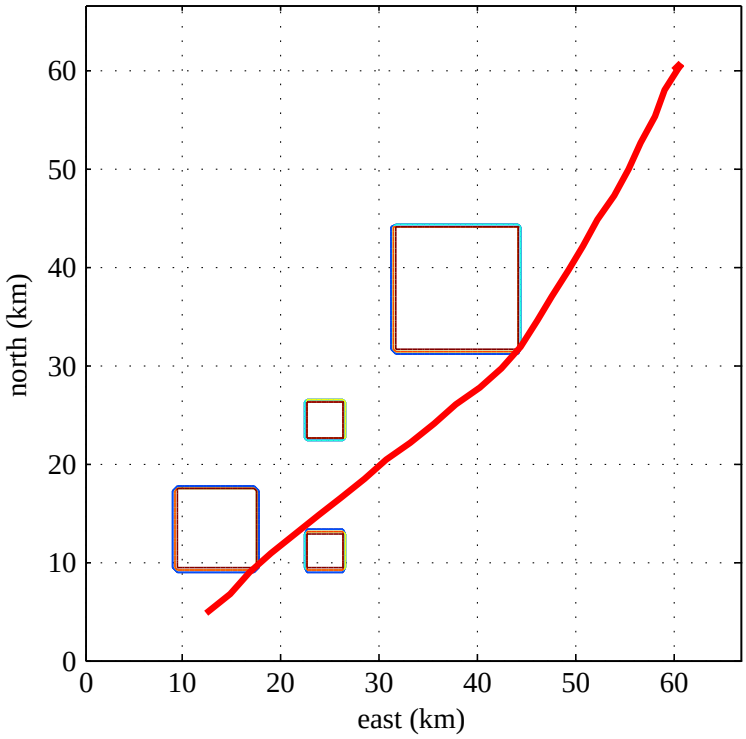


Figure 5.1. Path discovered by Kinematic Tree* Algorithm.

computed saving computation time. But this may lead to a path near a local minimum.

The effect of computational complexity for Kinematic Tree* algorithm has been discussed in length in Chapter 3. The error in the heuristic estimate of remaining cost to go grows with each node. Thus the algorithm runs in exponential complexity (See Chapter 3 for details when the KT* algorithm runs in linear complexity). The time it took to find the solution shown in Figure 5.1 took an hour in a Dell Intel Core i5 2.5 GHz machine. Simulation results for bounded heuristics results in very fast conversion (for e.g the case in which an oracle provides the optimal path). But for all practical applications a weighted heuristic function is used.

5.1.2 Varying the weights of g and h

The previous section has shown that Kinematic Tree* can compute optimal paths in static obstacle field. But as described in Chapter 3, the computational cost of achieving the optimal path is very high. This section examines the effect of weighting the two components of the cost function g and h in optimality and complexity of the algorithm.

Recall that the weighted evaluation function is given by

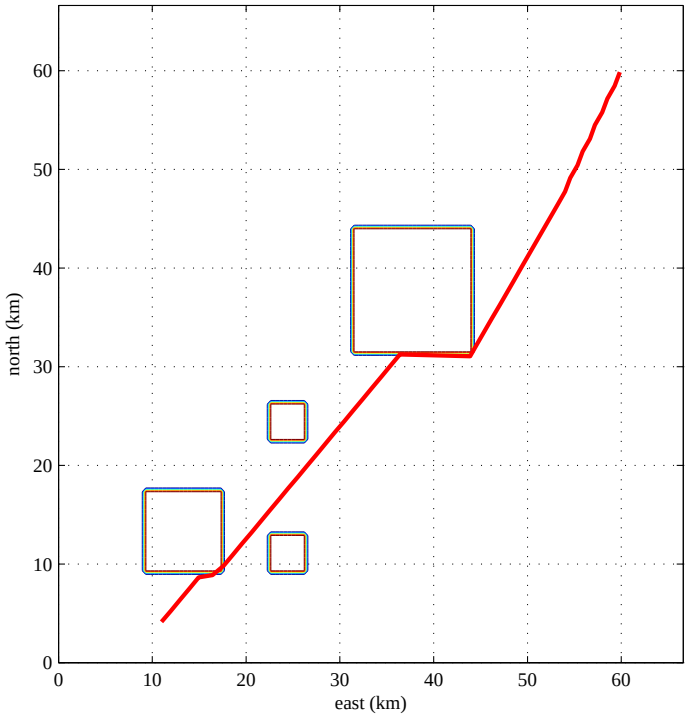


Figure 5.2. Path discovered by Kinematic Tree* Algorithm with $\alpha = 1$.

$$\bar{f}(x) = (1-\alpha)g(x) + \alpha h(x) \tag{5.8}$$

where, α is the weight parameter.

For $\alpha \leq 0.5$ the cost function is admissible and the the computational cost is similar to using the cost function $\hat{f} = \hat{g} + \hat{h}$. For $\alpha > 0.5$ the cost function loses admissibility and interesting observations can be made regarding optimality and computational complexity.

First, consider the case when $\alpha = 1$. Then the cost function is simply $\hat{f} = \hat{h}$. This means the algorithm minimizes the remaining cost to go. Very fast convergence of the algorithm is observed with evident suboptimal solution.

Figure 5.2 shows the paths discovered by the Kinematic Tree* algorithm for $\alpha = 1$. Evidently the planner has found a suboptimal solution (compare with Figure 5.1). As can be seen from the figure, the planner tries to minimize the remaining distance to the goal and only avoids obstacle when its immediate successors are in collision course with the obstacles (Figure 5.2).

This is a greedy algorithm and can be seen to follow the behavior of a *Bug algorithm* [76].

Table 5.1. Comparison of KT* algorithms for different values of α

α	**Distance (km)**	**Number of Nodes Expanded**	**Time Taken(sec)**
0.6	105.3	69257	54.00
0.7	105.3	5954	4.61
0.8	113.4	2739	2.24
0.9	121.8	1422	1.38

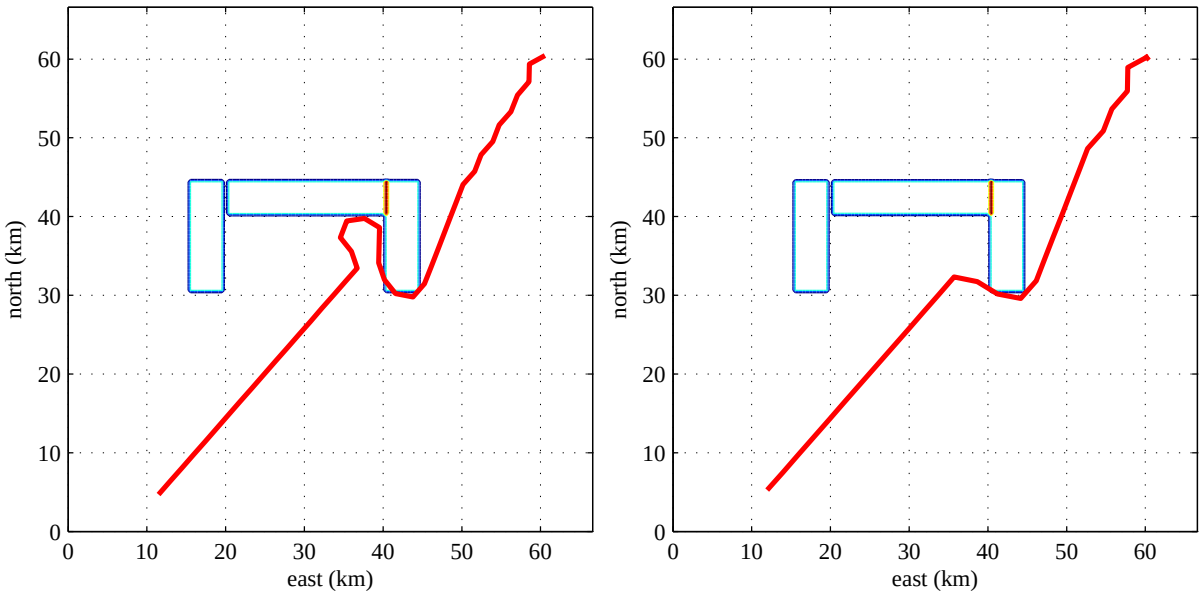
A Bug Algorithm replicates a behavior of bug movement. Bug behaviors are simple: (1) Follow a wall (right or left) and (2) Move in a straight line towards the goal. Bug algorithms fail to find a solution when there are concave obstacles in the scenario. One of the most important features of the Kinematic Tree* (as well as Kinematic Tree) algorithm is that all the branches are kept in the tree structure for possible future expansion and thus the Kinematic Tree* algorithm never remains stuck in a local minima.

A more challenging scenario arises when there is a local minima in the planning field. Figure 5.3 presents a scenario where there is evidently a local minima in the obstacle field. The horse shoe shaped obstacle field is located at $(15km \geq x \leq 20km, 30km \geq y \leq 45km)$, $(20km \geq x \leq 40km, 40km \geq y \leq 45km)$,and $(40km \geq x \leq 45km, 30km \geq y \leq 45km)$. The start point is (12 km ,8 km) and the goal is at (60 km,60 km).

Figure 5.3 shows the paths discovered by the Kinematic Tree algorithm for different values of α. As is evident from the figure path length increases with increasing value of α. One of the things to note is that none of the algorithms is ever stuck in the local minima. The algorithm always finds a path to the goal. In Chapter 3 the Kinematic Tree algorithm was proved to be complete, i.e. it is guaranteed to find a solution if a solution exits . This simulation results confirm that claim.

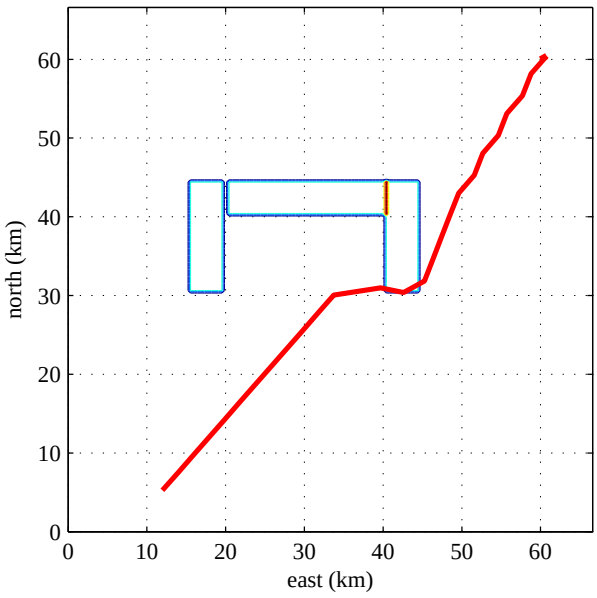
Table 5.1 summarizes the number of nodes expanded and the time taken to get a solution for different values of α. As expected as the quality of solution deteriorated (with increasing values of α), the algorithm converged quickly. The number of nodes expanded to get a solution also increase as the cost function approached being admissible. All simulations were carried on a Dell Dual Core 2.6 GHz machine.

Instead of keeping a fixed value of the weight a dynamic weighted Kinematic Tree* algorithm is explored in the next section.

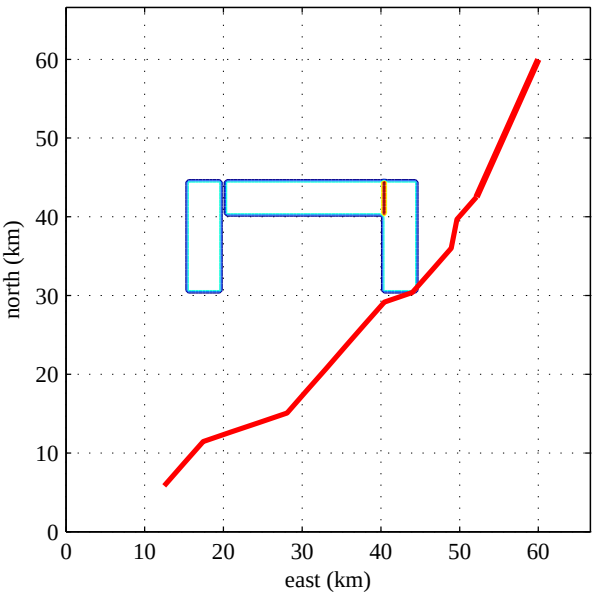


(a) $\alpha = 1$

(b) $\alpha = 0.8$



(c) $\alpha = 0.60$



(d) $\alpha = 0.5$

Figure 5.3. Comparison between the paths found by varying α.Upper Left: Path found by$\alpha = 1$; Upper Right: Path found by $\alpha = 0.8$. Down Left: Paths found by $\alpha = 0.6$. Down right: Paths found by $\alpha = 0.5$

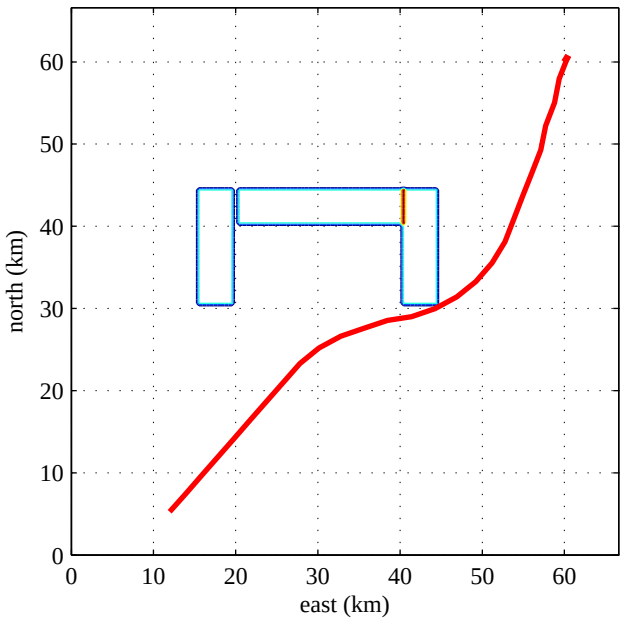


Figure 5.4. Path discovered by dynamic weighted Kinematic Tree Algorithm.

5.1.3 Dynamic Weighting

The Dynamic Weighted KT* was described in Section 5.1.3. This section shows the simulation results of a dynamic weighted KT* in an obstacle field. Recall the modified weighted cost function of the dynamically weighted Kinematic Tree algorithm is given by:

$$f(x) = g(x) + h(x) + \alpha \left\{ 1 - \frac{r(x)}{D} \right\} h(x) \tag{5.9}$$

Here $r(x)$ is the remaining distance to goal and D is the distance between the start and finish points. Thus the algorithm puts more weight on the heuristic at shallow levels of search and the cost function becomes admissible when the search reaches near the goal. It was shown in Chapter 3 that the dynamically weighted KT* algorithm can find solutions to the goal, where the bound on the cost can be predefined. Thus if C^* is the optimal cost, then the algorithm was proved to be find a solution that will converge with cost $(1+\alpha)C^*$

Figure 5.4 shows scenario. The start point is (10,10) and the destination is at (60,60). The obstacle is located at $(15 \geq x \leq 45, 30 \geq y \leq 45)$.

Figure 5.4 shows the paths discovered by the algorithm. As seen from the figure, the solution is suboptimal but better than most of the solutions found by constant α solutions as discussed in the previous section.

Thus all the theoretical results obtained in Chapter 3 have been verified through simulation results in this chapter. The claims about optimality and computational complexity have been substantiated with simulation results. The condition under which the Kinematic Tree* algorithm remains linear in computational complexity and at the same time provide optimal paths have been shown.

5.2 Path Planning on Mountainous Terrain

In this section path planning in a hilly terrain is considered. The difficulty in planning in such a scenario arises from the fact that the total configuration space cannot be divided binarily into either free or obstacles. The cost of reaching the goal (either distance traveled, time taken or energy expended) is dependent on the terrain. Further, the cost of transition is direction dependent: it is cheaper to travel downhill than uphill. This leads to a difficult optimal path planning problem.

The total energy expended by the vehicle is taken as the cost function that is minimized. The total energy expended by the vehicle is the work done against friction, which varies with terrain plus "hotel power" and energy to go uphill. Since friction is a non-conservative force work done is dependent on the path taken. This work done against friction is consistent and thus maintains the triangle inequality. This makes work done or energy expended an admissible cost function.

In [63] Jaillet et.al proposed a transition-based RRT planner, which computes low-cost paths that follow valleys and saddle points of the configuration-space costmap. Though no claims of optimality was done in that particular work. We use a similar minimum work cost path to prove optimal paths can be found by the $KinematicTree^*$ algorithm.

5.2.1 Vehicle kinematics and motion primitives

A point mass model is used to define the kinematics of the ground vehicle. The planar motion is dictated by an unicycle motion model while change in altitude is governed by terrain. The discretized model is same as Equation 5.1 and Equation 5.2

The energy required by the vehicle for traversing to a goal is a combination of the work done against friction along the path and the change in potential energy occurring due to terrain. For discretized path the total work done can be approximated as the the summation of work

done along each segment.

$$W = \sum_{i=0}^{n} \mathrm{F}_i \cdot \Delta \mathrm{s}_i \tag{5.10}$$

Ignoring the power required to accelerate to the desired velocity and assuming quasi-steady state along each transition, the work done for traversing a distance Δs in time Δt is simply power required times Δt.

In steady motion along a terrain of slope γ the power required by the ground vehicle moving with speed v_g against a non-conservative resistive force F_r and gravity is

$$P = (F_r + mg\sin\gamma)v_g + P_0 \tag{5.11}$$

where F_r is the total force needed to overcome dissipative forces acting on the vehicle, $mg\sin\gamma$ is the component of gravity force acting in the direction of vehicle motion and P_0 is the auxiliary power required (i.e. internal losses and power required to run ancillary equipment).

The total frictional forces acting on a car can be approximated by aerodynamic drag f_d and rolling friction f_{rr} between tires and terrain.

$$F_r = c_d v_g^2 + C_{rr} mg \tag{5.12}$$

where c_d is an aerodynamic drag constant (which is vehicle dependent) and C_{rr} the rolling friction coefficient (which is both vehicle and terrain dependent). Note that c_d is not a drag *coefficient*, since it is not a dimensionless quantity. Note that the rolling resistance is independent of velocity but the aerodynamic resistance is not.

Branches are precomputed on a flat level surface. Figure 5.5 shows the motion primitives computed at zero elevation. Each motion primitive is then projected onto terrain to determine the change in elevation that will result from travel along a branch. Average terrain slope along the m^{th} branch is

$$\tan\gamma_m = \frac{(z_m - z_0)}{\sqrt{(x_m - x_0)^2 + (y_m - y_0)^2}} \tag{5.13}$$

It is assumed that average terrain slope is small, so the difference in path length along the ground compared with precomputed path on flat ground is negligible. The energy required to travel the m^{th} branch at a speed v_g is thus

$$E_m = [(F_r + mg\sin\gamma_m)v_g + P_0]\Delta t \tag{5.14}$$

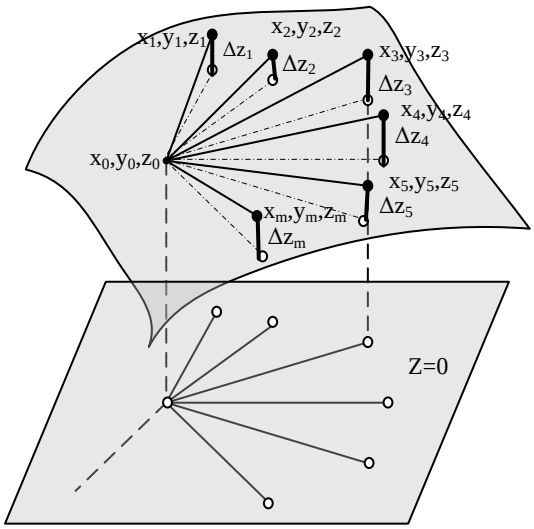


Figure 5.5. Motion Primitives calculated at zero elevation are projected onto the terrain.

5.2.2 Node selection and endgame region

Nodes are selected based on the heuristic cost function $f(n) = g(n) + \hat{h}(n)$ where $g(n)$ is the energy expended in the path to reach the node n and $\hat{h}(n)$ is an estimate of the expected energy required to reach the goal.

$$g(n) = \sum_{i=0}^{n} (F_{r,i} v_g \Delta t + P_0 \Delta t + mg\Delta z_i) \tag{5.15}$$

$$h(n) = F_{r,nom} v_{g,nom} \Delta t_{le} + P_0 \Delta t_{le} + mg\Delta z_{goal} \tag{5.16}$$

The expected cost $h(n)$ is calculated based on the energy estimates to drive at the difference of current elevation and goal elevation at nominal velocity. Δt_{le} is the estimate of time required to reach the goal traveling at nominal velocity. If the vehicle drives along a negative slope such that the component of its weight is more than the resistive forces then the power required to drive the vehicle is simply its auxiliary power.

$$P = max(P, P_0) \tag{5.17}$$

This heuristic cost function is admissible because it never overestimates the remaining cost

Table 5.2. Parameters for Ground Vehicle

variable	value	description
m	1800 kg	mass
c_{rr}	0.03	rolling resistance constant
c_d	1.392	air friction constant
P_0	10000 W	auxiliary power
P_{max}	200 kW	maximum power

to go (this is because the energy required to cover the remaining distance is always going to more than the straight line path with difference of slope). Thus our estimate to go is the optimal heurstic $\hat{h}(n)$ and $\hat{h}(n) \leq h(n) \forall n$.

The expected cost function $\hat{h}(n)$ assumes straight line path from current position to goal. This is different form the actual path discovered by KT* limited by discretization. The actual distance covered will be more than the straight line path due to discretization. To take into account the resolution error the expected cost is augmented by a factor of heading discretization. Recall that the heading is incremented by $\Delta\psi$. Thus the corrected expected cost is given by $\hat{h}(n) = \hat{h}(n)cos(\frac{\Delta\psi}{2})$. This results in faster convergence of the algorithm.

The end game region is simple euclidean distance to goal.

$$X_{goal} = \left\{ x : \frac{r_{goal}}{\delta t} \leq V_{nom} \right\} \tag{5.18}$$

5.2.3 Simulation results

As an example of hilly terrain consider central Pennsylvania. Paths are computed from different starting locations to a goal near the center of the map, and use a ground vehicle whose details are tabulated in Table 5.2.

From Equation 5.11, power consumed by the ground vehicles is given by

$$P = c_d v^3 + mgc_r v + P_0 \tag{5.19}$$

which gives,

$$\frac{P}{v} = c_d v^2 + mgc_r + \frac{P_0}{v} \tag{5.20}$$

Optimal speed to drive can be computed as

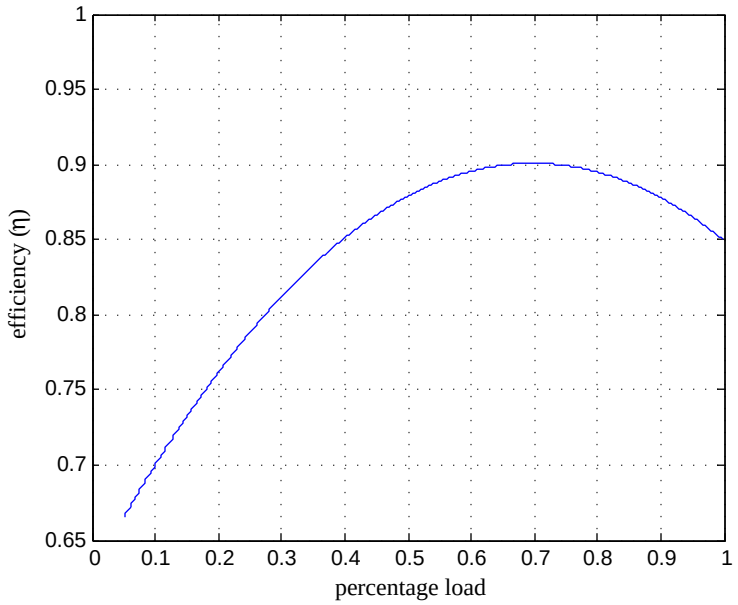


Figure 5.6. Efficiency as a function of percentage of load.

$$\frac{\delta}{\delta v}\left(\frac{P}{v}\right) = 0 \tag{5.21}$$

which gives,

$$v_{opt} = \left(\frac{P_o}{c_d}\right)^{\frac{1}{3}} \tag{5.22}$$

With the parameters given in Table 5.2, the energy required to travel on flat ground is 0.5 kWh/km at 25 m/s (equivalent to 5.6 L/100 km at 90 km/h, or 42 miles per gallon at 55 mph). Following Equation 5.22 the optimal speed to drive in flat surface is 15.4 m/s. But this is at ideal condition considering the entire energy in a battery is transfered to energy.

Efficiency of a typical electric motors varies with load. Figure 5.6 shows a typical efficiency curve of a electric motor with percentage of load. All electric motors have very good efficiency with slight variation with increasing load. The motor used for simulations has a maximum efficiency of 90% at two-thirds of maximum load.

Maximum power of a typical car is around 100KW (140 hp approx). A further constraint

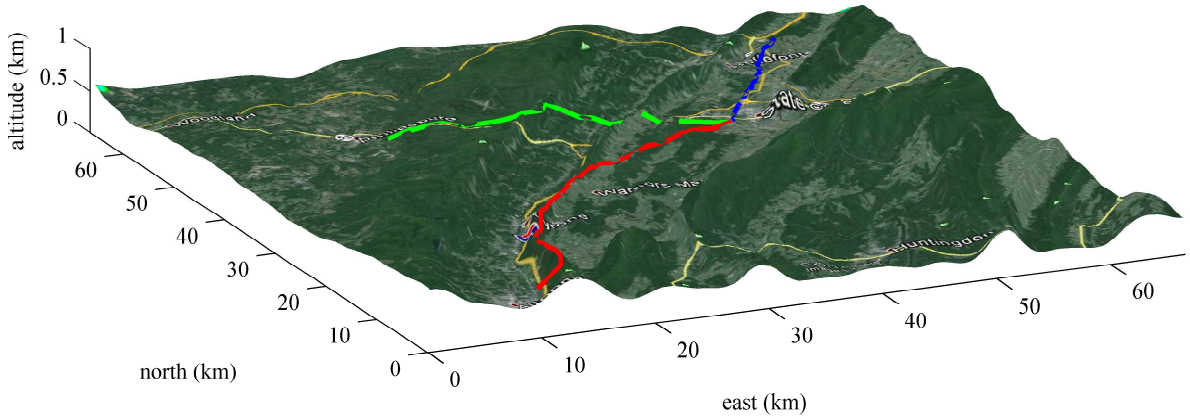


Figure 5.7. Minimum energy Path for Ground vehicle over Central Pennsylvania.

Table 5.3. Ground vehicle path costs, distances traveled and time taken to reach the destination for both energy-optimal path and straight line path.

	straight line path			**energy-optimal path**		
start	**distance**	**cost**	**time**	**distance**	**cost**	**time**
	km	kWh	minutes	km	kWh	minutes
1 (red)	51.69	31.25	37	55.06	30.45	39
2 (green)	30.29	18.09	20	32.34	17.69	24
3 (blue)	23.02	13.35	15	23.03	13.16	15

of maximum power of 100KW was considered. This means the vehicle to travel a steep slope its velocity has to be decreased. The results presented in this section uses this ground vehicle model.

In this example the terrain type is assumed to be constant (so that the coefficient of rolling friction is constant), although that is not necessary for this path planner.

Figure 5.7 shows the paths found by *Kinematic Tree** algorithm for different start locations. Three different starting points were chosen randomly and the goal was in the center of the graph (representative of State College region in central Pennsylvania.) The path shown in *red* follow the ridges for most of its path towards the goal. The path shown in *green* navigates between terrain to find the energy optimal path. The path shown in *blue* is a fairly straight line path to the goal.

Distances traveled and path costs for three start positions are shown in Table 5.3. As a

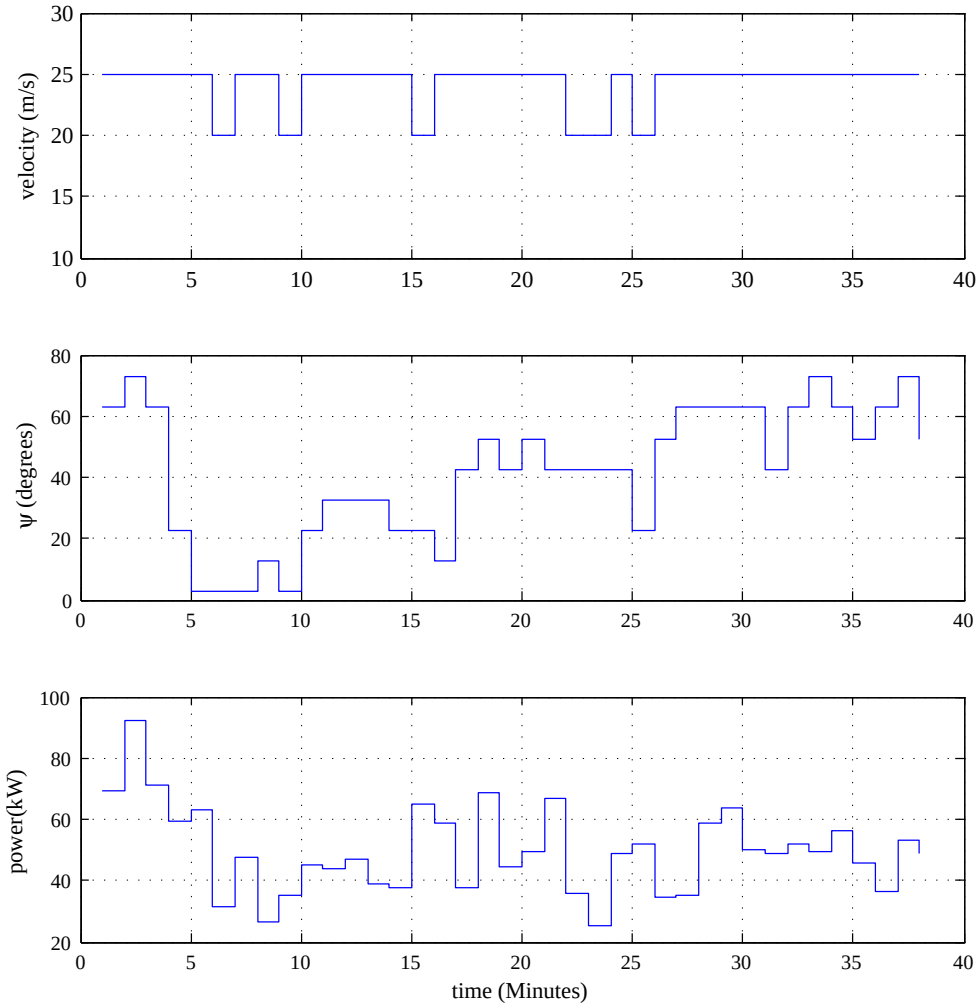


Figure 5.8. Velocity, heading and power consumed for the red path.

comparison, the total energy consumed for the straight line path is considered. For the straight line path the vehicle had to climb steep slopes and thus its velocity came down to respect the maximum power constraint of the car. Thus the time to reach the destination for the straight line path was very similar to the energy-optimal path though the distance traveled was smaller than the energy optimal path (Table 5.3).

Table 5.3 table compares the distance traveled for the energy optimal path and straight line path and their corresponding energy consumption. It is seen that though the energy optimal path is longer than the straight line path by almost 10% the energy consumed by the energy

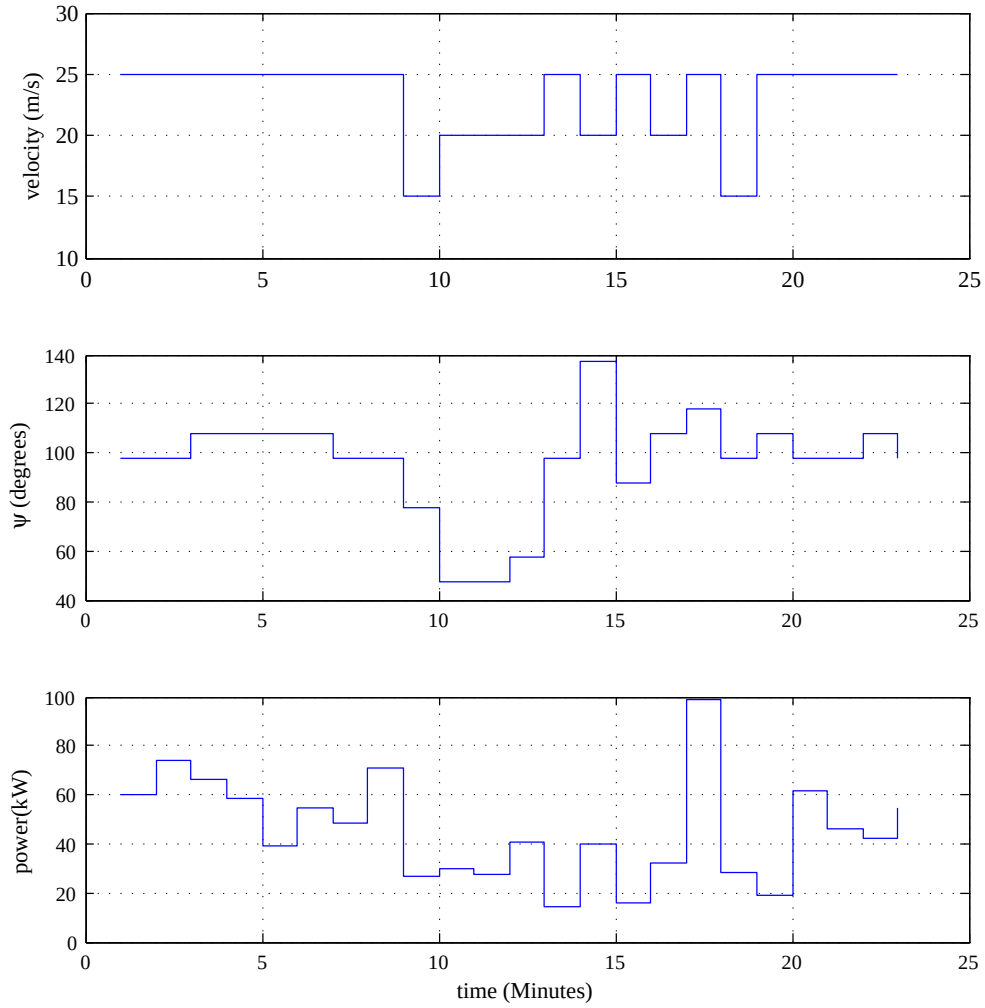


Figure 5.9. Velocity, heading and power consumed for the green path.

optimal path was 3% less than the straight line path. As noted before, in the straight line path the velocity was low and thus to reach the goal the time required was similar to that of the energy optimal path, though the distance covered in straight line path was considerably small. Thus the energy optimal path was not only beneficial in terms of energy consumed but also resulted in paths that was similar in terms of time it takes to reach the goal.

Similar results were obtained for the other starting positions as well. Once the path climbs to an altitude it generally tries to maintain the potential energy gained and thus follow the ridges.

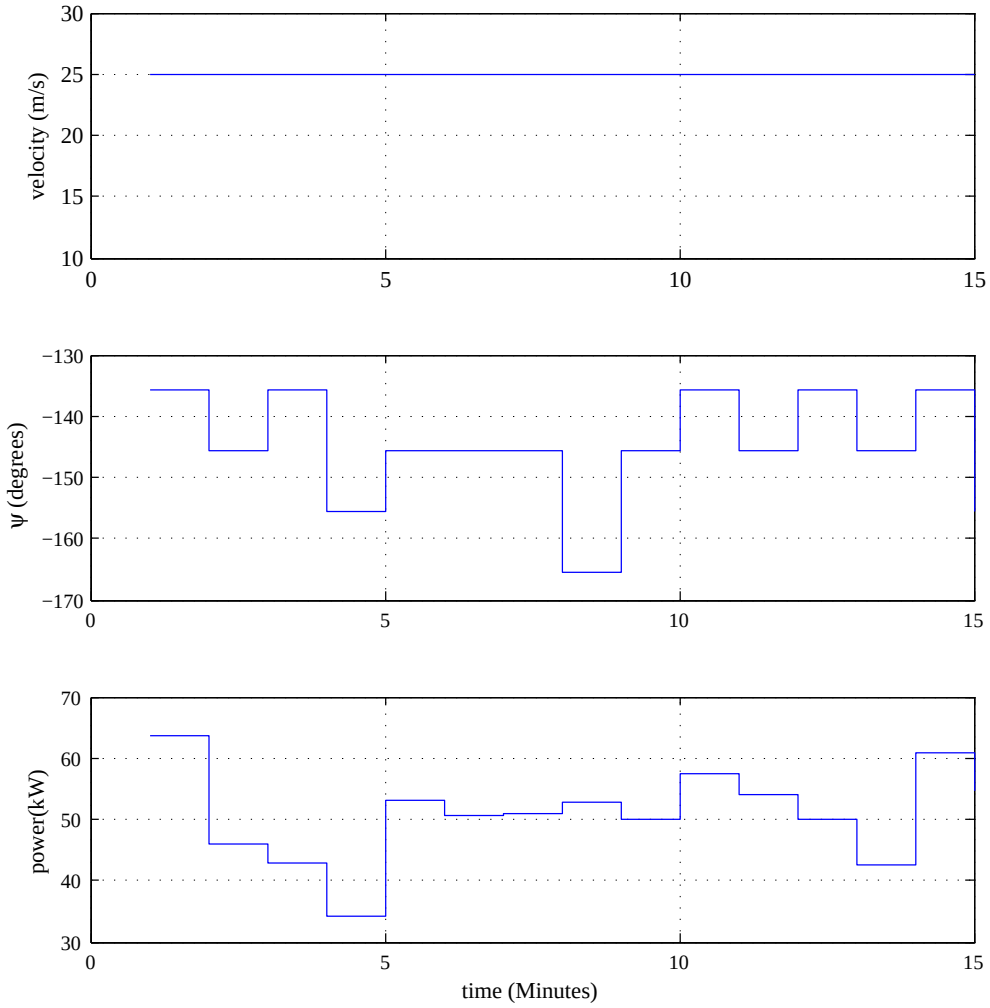


Figure 5.10. Velocity, heading and power consumed for the blue path.

Figure 5.8 shows the commanded velocity and heading changes for each segment of the *red* path. Corresponding power consumed at each segment is also plotted. The energy consumed is grater when the vehicle had to climb a steep hill and power reduced while going downhill. Note the maximum power restriction of 100 KW was never violated in any segment of the path.

Figure 5.9 and Figure 5.10 shows the corresponding plots of velocity, heading and power consumed for the *green* and *blue* paths. The alternate steep slopes in the *green* path is reflected with change of speed and power along the path. Similar results can be seen for *blue* path as well.

Thus the Kinematic Tree algorithm successfully finds energy optimal paths in mountainous terrain. Note that coefficient of friction is kept constant in the simulations. Traversibility of the terrain can be incorporated easily in each branch of the Kinematic Tree in a similar manner in which the terrain information is incorporated in the planner. Thus the Kinematic Tree* framework has been shown to be a great tool to find energy optimal paths in mountainous terrain.

5.3 Summary

In summary this chapter has shown simulation results for the Kinematic Tree* algorithm for ground robots. Different applications were:

- The Kinematic Tree* algorithm was shown to find optimal paths in a obstacle field.
- A weighted Kinematic Tree which uses a weighted cost function to escape local minima and provide quick convergence was also discussed.
- The Kinematic Tree* algorithm was used to find minimum energy paths in mountainous terrains of Central Pennsylvania.

This chapter has shown that the algorithms developed for path planning of unmanned aerial vehicles to explicitly handle time varying winds can be used for ground robots in difficult planning problem. Hence this thesis has shown a comprehensive planning framework which takes into account the surrounding environment, wind in case of an UAV and terrain in case of ground vehicles.

Chapter 6

Conclusion

Harvesting energy from the atmosphere is a challenging task. Small UAVs, used for many applications like surveillance and search and rescue, are restricted in their operations because of energy constraints. Wind is a naturally occurring phenomena which if properly utilized can lead to significant increase of range of operations of current UAVs.

The research presented in this dissertation describes a means to exploit atmospheric energy to increase range and duration of flights of small UAVs. The research is motivated by flight of birds. Large birds which are of the same size as small UAVs have developed techniques to fly for hundreds of kilometers without using their their internal energy. They use atmospheric energy to fly to their destination and thus follow only specific routes to reach the goal.

This dissertation focused on path planning for unmanned aerial vehicles to exploit atmospheric energy to maximize range and endurance of small unmanned aerial vehicles. Path planning in a flow field is a extremely challenging problem as the configuration space is a time dependent function which affects the vehicle state at all the time.

Earlier research focused on graph based planning approach. Graph based planning approach works really well when fixed way points are given a priori. The cost of transition can be optimized for each segment. The weighted directed graph then can be searched to find energy optimal routes. Prior research has shown a variation of breadth fast search called the *Energy Map* works very well for static wind fields. A practical application of the Energy Map techniques was used in the Green Flight Challenge and the details are presented in Appendix A.

The dissertation presented a kinematic based algorithm to handle path planning problem in complex wind conditions. Planning in presence of time varying three dimensional wind

is a challenging problem due to the complex nature of the wind fields. Also the temporal variation of the wind adds to the complexity. Standard graph based motion planners as well as randomized motion planning techniques fail to provide feasible solution in this complex environment.

The Kinematic tree algorithm was introduced to handle this path planning problem. The kinematic tree algorithm was shown to be resolution complete, i.e, it was able to find a solution to the problem if a solution exits. The algorithm was shown to be computationally efficient compared with standard randomized planner. The algorithm can also be extended to be optimal if the cost function is admissible.

The algorithm was tested in time varying complex wind data over central Pennsylvania. Flight to a distance goal was considered which was considerably out of the gliding range of an un-powered aircraft. The method was shown to be effective in identifying feasible paths and also computed the feasible starting times during the day to make a flight to the goal possible.

The Kinematic Tree algorithm is extended to handle path planning for ground robots. Path planning in mountainous terrain is considered to show the efficacy of the method in complex terrain.

6.1 Summary of Contributions

6.1.1 Kinematic Tree Algorithm

Though graph based planning method was suitable for 2-D static wind field the extension to 3-D dynamic wind fields proved problematic. Sampling based motion planners are used in robotics community to handle the curse of dimensions. Sampling based algorithms, though are very successful in finding paths for unmanned systems there is no guarantee of optimality.

The problem in hand was to identify energy efficient routes and thus the quality of paths discovered is of prime importance. To effectively handle 3-D time varying winds Kinematic tree algorithm was introduced which was based on kinematics of the unmanned system. Time was inherently embedded in the tree structure which enabled the algorithm to handle time varying complex flow field.

Some of the properties of the Kinematic Tree which are important for implementation in real application like completeness, and computational complexity are discussed in details. The algorithm is proved to be resolution complete; this means that given the resolution parameter

is tuned the algorithm is guaranteed to find a solution if it exists. A lot of computational complexity is solved by pre-computation of motion primitives of the algorithm. The algorithm is shown to be computationally more efficient that the Resolution complete RRT algorithm for this particular application.

Admissibility of cost function cannot be guaranteed for harvesting energy from the atmosphere. Thus only energy aware paths can be found for energy harvesting application. For path planning for ground robots the Kinematic Tree algorithm provides optimal paths.

6.1.2 Kinematic Tree* Algorithm

The optimal version of the Kinematic Tree algorithm named the Kinematic Tree* was introduced for planning for ground robots. For ground robots the cost function that is minimized is typically distance covered. Distance is an admissible cost function. Moreover, in general, energy expended by a ground vehicle is also admissible. Thus optimal path planners can be designed for ground robots. The Kinematic Tree* algorithm was shown to find optimal path for obstacle avoidance and for planning in mountainous terrain.

Often there is explicit trade between optimality and computational complexity. Detailed theoretical insight was provided on the relations between heuristics for optimality and how that effects the optimality.Theoretical results were verified through simulation results. The effect of variation of heuristics with optimality and computational complexity was analyzed through simulation results.

One of the main results that was discussed is that if there is unbounded error in the heuristic then the complexity of Kinematic Tree* algorithm becomes exponential. Thus for practical applications a good initial guess of cost-to-go is essential to bound polynomial complexity of the algorithm.

6.1.3 Environment aware planning

The simulation results showed the efficacy of planning in complicated 3-D time varying wind field. The Kinematic Tree algorithm was able to find energy efficient routes over central Pennsylvania. The algorithm successfully found time to launch and paths to follow to reach a destination by using energy only from the atmosphere.

The same algorithm when applied for ground robots finds optimal paths in obstacle field. Minimum energy paths in mountainous terrain can also be identified by the algorithm.

Thus being aware of the environment planning in much more efficient ways can be made for autonomous system. The Kinematic Tree makes autonomous system much more efficient in reacting to surrounding environment and thus making better decisions and find intelligent paths.

A unified framework to handle a difficult planning scenario is the main contribution of this dissertation. Results for Kinematic Tree planner for both air and ground vehicles are presented in this dissertation. When the cost function is admissible the planner returns optimal path. The algorithm has the potential to be used for many practical applications.

6.2 Future Works

The method developed in this dissertation holds promise to be used for many practical applications. As small UAVs are being increasingly used for many applications, the need for planning, considering environmental factors, becomes increasingly important. The speed at which these vehicles operate is very similar to that of wind speeds. Thus consideration of wind information is very important for small UAVs.

The general method of planning described in this dissertation can be adopted and modified for many possible applications. Some of the possible application areas are identified below.

6.2.1 Unknown wind field

In this research planning was done based on given wind field. For the Green Flight Challenge planning was done based on prediction of wind field from the meteorological department. Though most of planning problems are done in known environments for real life applications extension of the algorithm for incorporating realtime winds has to be considered.

Planning for correct paths requires precise measurement of wind. In [77] Quindlen uses a five hole probe to measure wind vector magnitude and direction. Integration of this technique with the planner will result in real time planning in unknown environment.

This will be critical in future missions where real time data assimilation is critical especially in potential hazardous environments.

6.2.2 Applications

The methods developed in this research has potential applications for various fields. The planning framework developed here has the potential use in many science applications.

6.2.2.1 Science Missions

Hurricanes are one of the most destructive natural phenomenon on Earth. Understanding and predicting the dynamic behaviors of cyclones and super cell thunderstorms remains one of the Grand Challenges in 21st century. Unmanned Aerial Vehicles are useful autonomous platforms that can actively assimilate and explore in places that are too hostile for crewed aircrafts. In [78] Frew discusses the challenges that must be addressed if small unmanned aircraft are to be used in this application.

The NASA Global Hawk aircraft are ideal platforms for investigations of hurricanes, capable of flight altitudes greater than 55,000 ft and flight durations of up to 30 hours. Global Hawks extensive instruments include scanning high-resolution Interferometer Sounder, dropsondes, theTWiLiTE Doppler wind lidar, and the Cloud Physics Lidar, while the over-storm payload includes the HIWRAP conically scanning Doppler radar, the HIRAD multi-frequency interferometric radiometer. These instruments can provide high resolution 3-D wind information [79].

In [80] Mohseni reports that MAVs have much greater control over their altitude than horizontal motion inside and hurricane and they have developed controllers to exploit areas and volume coverage of hurricane eye and eyewall region. In [81] Elston et al. reports results from field deployments of the Tempest Unmanned Aircraft System, to perform in situ sampling of supercell thunderstorms, including those that produce tornadoes.

The wind information provided by Global Hawk can be incorporated to guide a small UAV in a hurricane to measure *in situ* data. Critical planning is needed to fly UAVs in ”safe” regions inside an hurricane and it requires 3-D planning in time varying wind fields. The Kinematic Tree [82] have shown its capabilities in handling 3-D wind fields in relatively complex environment. This planner can be modified to handle safe regions to fly inside such adverse conditions.

6.2.2.2 Next Generation Air Transportation

Path planning and optimization in flow field can also have potential application in other research areas like air transportation.

Some of the key ares of focus for next generation Air Transportation System are safety of air traffic controllers, ground efficiency with accurate takeoff and landing schedules and reduce weather and congestion delays. Path planning for air transport transport systems to increase fuel efficiency is one of the possible future areas for research.

The Air-Force Research Lab and DARPA have been looking into the challenges and opportunities associated with flying aircrafts in formation under Surfing Aircraft Vortices for Energy (SAVE) concept. In the paper [83] they report a a fuel efficiency of around 10 % for flying C-17 aircrafts in formation.

A different approach can be taken. Saving fuel consumed by aircraft by optimizing speed to fly and power settings based on local wind conditions can be considered for next generation air transport system.

Planning methods based on wind routing has been considered for both crewed and uncrewed aircraft. Rubio [32] has used a genetic algorithm, while Jardin used a method based on neighboring optimal control [33]. Both of these methods shows how the planning system makes use of favorable winds for fuel consumption benefit.

6.2.2.3 Autonomous Rovers

A lot of advancement have been made in autonomous car technology. Future autonomous ground vehicles, specially the ones employed in challenging environments will need to asses the traversibility of paths. The Kinematic Tree* algorithm developed in this research has potential to be extended to handle planning in rough terrain.

Appendix A

Green Flight Challenge

This section presents the test results for the Green Flight Challenge. The Green Flight Challenge (GFC) was organized by NASA to spur extreme flight efficiency for general aviation aircraft. The qualifying standard for the GFC was an aircraft that could fly 200 miles at an average groundspeed of 100 mph at a fuel consumption of 200 passenger miles per gallon. The GFC qualifying fuel efficiency was better than that of current general aviation aircraft by a factor of 2 or more, and it is in fact equivalent to the fuel consumption of a Toyota Prius at nearly twice the speed.

In Green Flight Challenge a graph based planning algorithm was developed to minimize the fuel consumption of our aircraft the Taurus G4.

The Taurus G4 aircraft is based on the Pipistrel Taurus, a two-seat (side by side) self-launching sailplane. Its most distinguishing feature is a twin fuselage design as shown in Figure A.1.

A.1 Performance Data of Taurus G4

The data plotted in Figure A.2 were obtained from analysis and tuned using flight test data, with a focus on matching measured performance at speeds ranging from 90 miles per hour to 110 miles per hour true air speed. Polynomial curve fits of the data were used to compute energy minimizing flight conditions using gradient-descent optimization:

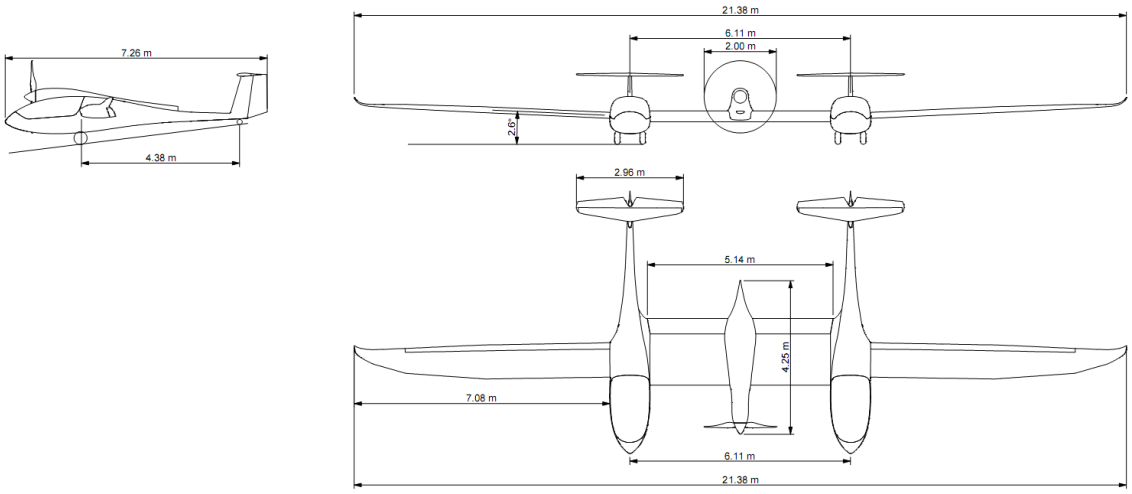
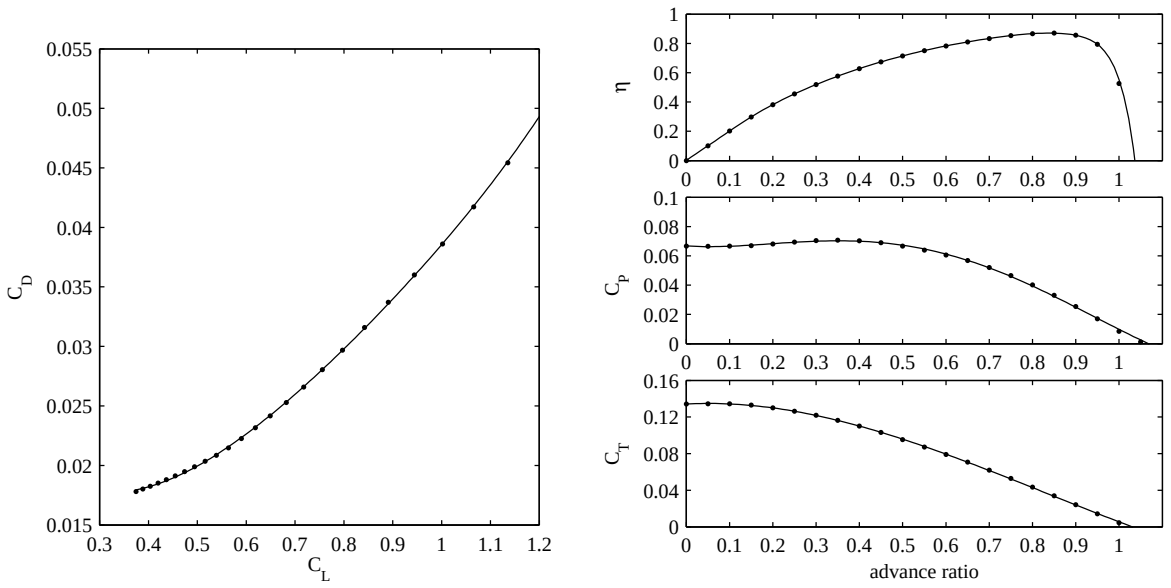


Figure A.1. External dimensions and configuration of the Taurus G4

Table A.1. Taurus G4 basic data[5]

Sizing	
Length	24.3 ft (7.4 m)
Wing span	70 ft (21.36 m)
Wing area	232 ft^2 (21.6 m^2)
Empty weight (ex. batteries)	1393 lb (632 kg)
Empty weight (in. batteries)	2495 lb (1132 kg)
MTOW	3307 lb (1500 kg)
Competition weight	3298 lb (1496 kg)
Performance	
Stall speed	51 mph (82 km/h)
Cruise speed	100 mph to 125 mph (160 to 201 km/h)
V_{NE}	135 mph (217 km/h)
Take off (over 50 ft (15 m) obstacle)	1970 ft (600 m)
Climb rate	885 fpm (4.5 m/s)
L/D at 100mph	28:1
Endurance	$>$ 2h 45 min
Range	281 miles (450 km)
Properties	
C_P	$0.2075J^4 - 0.4700J^3 + 0.2261J^2 - 0.0205J + 0.0668$
C_T	$0.0043J^4 + 0.0946J^3 - 0.2534J^2 + 0.0259J + 0.1341$
C_D	$0.0395C_L^4 - 0.1336C_L^3 + 0.1876C_L^2 - 0.0844C_L + 0.0295$



(a) Aircraft drag polar

(b) Propeller performance

Figure A.2. Aircraft performance data and curve fits for drag polar (left) and propeller (right) . Data is shown as points, the fitted curve curve is shown as solid line.

A.2 Graph Based Planning and the Green Flight Challenge

In this section graph based planning technique will be discussed in context of long range planning in a complex wind field. The application of graph based planning will be shown in Green flight challenge. Finally the shortcomings of graph based planning techniques will be discussed.

Optimal speed to fly between nodes and graph based path planning has been discussed in details in my masters thesis [56]. In this chapter graph based planning in the context of Green Flight Challenge is discussed. In the Green Flight Challenge a sequence of way points were given a priori. A *planning graph* was constructed based of feasibility of transition and was searched to find optimal trajectory.

The limitations of graph based planning is outlined and the need for sampling based planning is shown.

A.3 Graph Based Planning Technique

Graph based planning techniques for finding energy optimal path essentially requires a search of a weighted directed graph. First the environment is discretized with nodes and it is assumed that the vehicle is capable of translation between the nodes. Thus the continuous problem of finding a path from start to goal is reduced to finding a sequence in a gridded environment. The cost of transition between adjacent nodes is the weight assigned for the transition.

To plan for a path in a wind field the cost of transition between adjacent cells involves computing cost of transition through the wind. This cost can be optimized at each step and thus a step by step optimized path is possible in this approach. My earlier research [56] has shown that the Energy Map approach works very well compared to other methods. Essentially the Energy Map is a *planning graph* which is a powerful data structure that encodes information about which states may be reachable. A modified approach was taken for the Green Flight Challenge.

A.4 The Competition

The GFC course consisted of four laps flown counter clockwise around a roughly triangular course (Figure A.3 and Table A.2). There were a total of 26 legs in the course. The longest leg is from pin F to pin B, a distance of 17 miles; the leg from pin S to pin A runs from the button of STS Runway 19 to the end of Runway 19. Note that takeoff and the landing pattern were not included in time or distance flown and the energy computation. The aircraft was required to reach 4000' MSL by 17 miles into the course (at the pin labelled E) and had to remain between 4000' MSL and 6500' MSL until the last turn onto the leg from pin F to pin B. The point to point distance of the course was 185 miles; actual distance flown was computed from a GPS trace computed using equipment provided by the CAFE Foundation. This removed the influence of aircraft turn performance on score.

We took part in the competition with the Pipistrel 'sTaurus G-4 aircraft. The details of the aircraft are presented in Appendix A. Before setting up the grid for planning for the Green Flight Challenge the optimization problem is presented. The optimization problem was discussed in the previous work [56]. The next section discusses the optimization problem specific to the Taurus G4 aircraft.

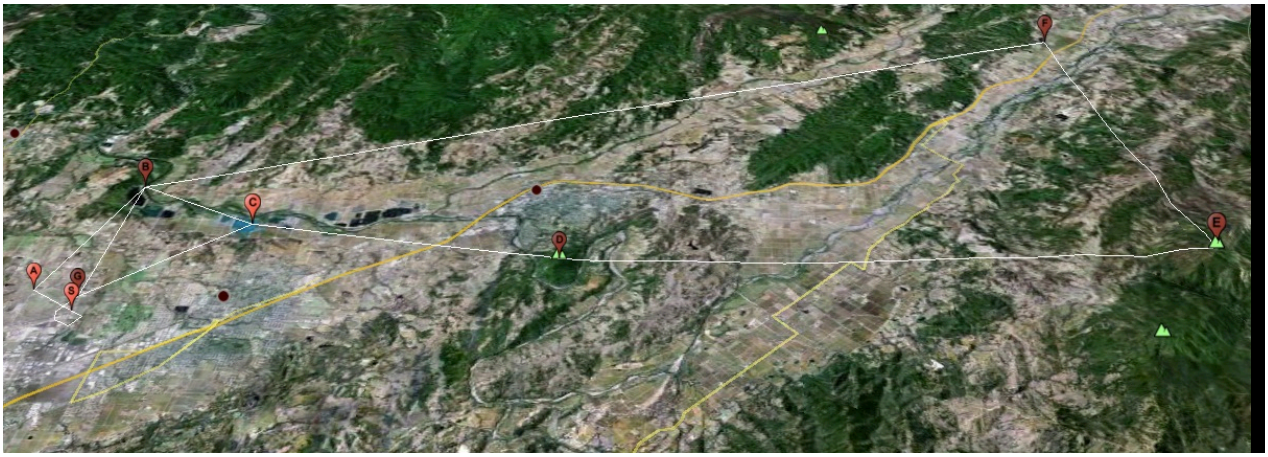


Figure A.3. Green Flight Challenge course looking west. Paddles show GPS waypoints; brake release occurs at the paddle labelled "S;" the finish line is at the paddle labelled "G."

Table A.2. Turnpoint locations and sequence

Waypoint ID	**Name**	**latitude** north	**longitude** west	**turnpoint**
S	start	38.514414°	122.812975°	brake release
A	Rwy 19 end	38.503161°	122.820953°	1
B	River Bend	38.513928°	122.869406°	2, 7, 13, 19, 25
C	Pond	38.544531°	122.853031°	3, 9, 15, 21
D	Fitch Mountain	38.618470°	122.840712°	4, 10, 16, 22
E	Geyser Peak	38.764572°	122.845345°	5, 11, 17, 23
F	101 Reservoir	38.748069°	122.972378°	6, 12, 18, 24
G	CAFE	38.513827°	122.818918°	8, 14, 20, finish

A.5 Flight Between Nodes

The power and speed to fly to minimize energy consumed for a given distance travelled and a specified climb rate can depend strongly on wind. Following Chakrabarty and Langelaan[2], winds are first decomposed into cross-wind (w_c), tail-wind (w_t) and vertical components (w_z) (Figure A.4).

Ground speed (v_g)and the required heading (ψ_g) to maintain the desired ground track for a given airspeed (v_a) are

$$v_g = \sqrt{v_a^2 \cos^2 \gamma - w_c^2} + w_t \tag{A.1}$$

$$\psi_g = \psi + \sin^{-1} \frac{w_c}{v_a} \tag{A.2}$$

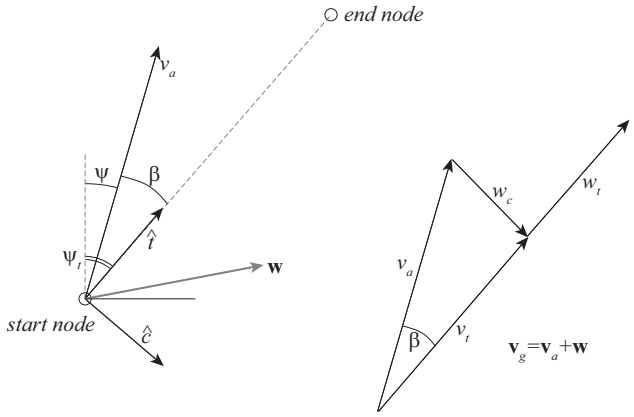


Figure A.4. Track coordinate frames (left) and resolution of airspeed and wind vectors into the track coordinate frame (right). Positive angles are shown.

In addition to the ground track constraint ψ_g, the flight path is also constrained by the ground-relative flight path angle γ_g.

$$\gamma_g = \tan^{-1} \frac{\dot{h}}{v_g} \tag{A.3}$$

where

$$\dot{h} = v_a \sin\gamma + w_z = v_a \frac{T-D}{mg} + w_z = v_a \frac{qS}{mg}(C_T - C_D) + w_z \tag{A.4}$$

represents the rate of change of altitude. T and D represent thrust and drag and C_T and C_D their corresponding coefficients; q is dynamic pressure; S is wing area; m is aircraft mass; g is acceleration due to gravity.

The required flight path angle is a function of the segment length and the required altitude change:

$$\gamma_g^{req} = \tan^{-1} \frac{\Delta h}{\Delta s} \tag{A.5}$$

From the standpoint of energy efficient flight the critical aircraft parameters are drag, power and thrust. Figure A.2 shows propeller performance curves and aircraft drag coefficient as a function of lift coefficient. Propeller data are given as a function of advance ratio J, where

$$J = \frac{v_a}{nD} \tag{A.6}$$

Power coefficient and thrust coefficient are[84]

$$C_P = \frac{P}{\rho n^3 D^5} \tag{A.7}$$

$$C_T = \frac{T}{\rho n^2 D^4} \tag{A.8}$$

where, P is the power required, D magnitude of drag force, and n propeller speed in revolutions per second. Note that the C_P is propeller power coefficient, and thus includes propeller efficiency.

In steady state flight,

$$\frac{\Delta E}{\Delta s} = \frac{dE}{dt}\frac{dt}{ds} = \frac{P}{v_g} \tag{A.9}$$

The energy consumed per distance travelled is (Equation A.9 and A.7, and including the efficiency of energy conversion):

$$\frac{P}{v_g} = \frac{C_P \rho n^3 d^5}{\eta_{ec} v_g} \tag{A.10}$$

The flight condition that minimizes energy consumed for distance travelled in arbitrary winds and satisfies flight path requirements can thus be computed by solving the constrained optimization problem

$$\text{minimize } \frac{C_P \rho n^3 d^5}{\eta_{ec} v_g} \tag{A.11}$$

$$\text{subject to } C_P = f_P(J) \tag{A.12}$$

$$C_T = f_T(J) \tag{A.13}$$

$$0.01 \leq J \leq 1.5 \tag{A.14}$$

$$\gamma_g = \gamma_g^{req} \tag{A.15}$$

$$v_g \geq v_{g,min} \tag{A.16}$$

For the Taurus G4 the polynomial functions f_P and f_T (which define power coefficient and thrust coefficient, respectively, in terms of advance ratio) are given in Table A.1.

Often the minimum ground speed will be zero (ensuring only that progress towards the goal is made), but sometimes greater minimum ground speed must be observed. In the Green Flight Challenge, for example, the minimum average ground speed was 100 miles per hour

(44.7 m/s).

Note that for level flight in zero wind, with $v_{g,min} = 0$ the optimization problem simplifies to

$$\text{minimize } \frac{P}{\eta_{ec} v_a} \tag{A.17}$$

$$\text{subject to } \gamma_g = 0 \tag{A.18}$$

where $\frac{P}{\eta_{ec} v_a} = \frac{D}{\eta_{ec}}$. The solution of energy-optimal flight for the zero wind case is thus flight at minimum drag, which is the well-known solution for range-maximizing flight for propeller-powered aircraft[84].

A.6 Flight Planning

In general winds do not vary monotonically or particularly smoothly with altitude; for example a shear layer can cause a very fast change in both wind speed and direction as altitude changes. A straightforward gradient-based trajectory optimizer is thus unlikely to find a global optimum. A similar problem exists when planning flights through cluttered environments: as part of minimizing distance flown one must decide to fly to the left or right (or over or under) obstacles. A genetic algorithm for planning in complex wind fields is described by Rubio[85]. Techniques such as mixed integer linear programming (MILP) have been used to address this problem[86, 87], and a mixed approach is also used here. First a segment by segment optimal trajectory was computed using a graph-based approach; this trajectory was then refined using a gradient descent optimization. The resulting flight plan defined speed to fly, power, climb rate and heading on each segment of the course. While it was possible to compute flight plans at higher spatial resolution (so that the speed, power, climb rate and heading would vary along a segment) this did not appreciably improve predicted energy consumption and would have resulted in increased pilot workload.

A.6.1 Segment by segment optimization

The course is discretized by defining a set of ten nodes at each of the 26 turnpoints around the course (note that a turnpoint denotes a specific turn around a waypoint: as shown in Table A.2 most waypoints are used as turnpoints multiple times). These nodes were placed at

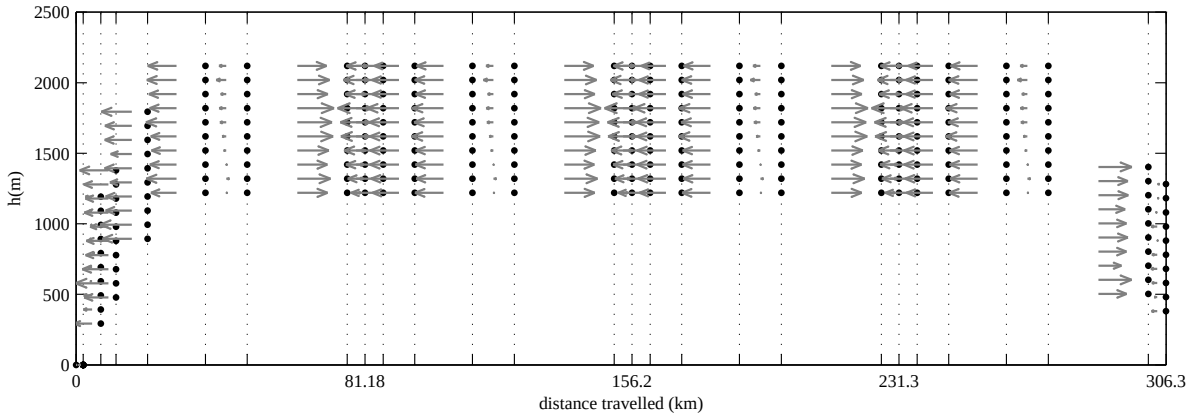


Figure A.5. Node placement and in track winds in the course.

100m increments in altitude with the lowest placed at the minimum allowable altitude for that turnpoint. The winds at each node were obtained from the high resolution wind field by interpolation. Figure A.5 shows the course "unwrapped," with total distance along the course as the horizontal axis and altitude as the vertical axis. The point to point distance around the course is 185 miles (298 km); to reflect the extra distance flown around each turn 330 m was added to the length of each segment. Winds for September 27 at 1100 PDT are shown for midpoints of each segment, with tailwind pointing to the right: see Figure A.12 for a vector plot of the horizontal wind field at 4000' MSL (1219 m MSL).

This set of nodes $i = [1 \ldots N]$ (with $i = 1$ defining the start and $i = N$ defining the goal) and edges ij that connect neighboring nodes i and j define all possible paths over the course (Figure A.6).

On each edge the wind is assumed to be constant, and is computed as the average of the wind at the start node and destination node of that edge. This results in some loss of resolution, but it allows the energy cost of a particular flight condition (power and airspeed) to be computed in closed form for a segment. This greatly speeds up the computation of optimal flight condition.

For each allowable edge ij the minimum cost flight condition is computed using equations A.11 through A.16. If flight over an edge is infeasible (i.e. it is outside the aircraft's performance envelope) it is removed from the set of allowable edges; a feasible edge must also bring the aircraft closer to the goal. For an edge, the computation of optimal flight condition defines the airspeed and power for that segment of the flight. The energy required, heading, and climb

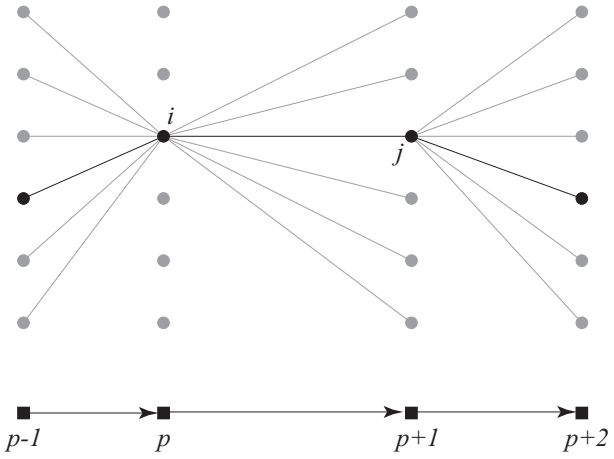


Figure A.6. Turnpoints (squares) and nodes (circles) used for flight planning.

rate for that segment are computed as part of the optimization; the required energy e_{ij} to fly from node i to node j defines the cost of the segment.

The set of edges and costs e_{ij} defines a directed graph to the goal. A flight plan π_k defines a particular path through the graph, and the cost of the plan is computed as the sum of the energies required to fly each segment (i.e. the sum of the edge costs to the goal):

$$E(\pi_k) = \sum_{i \in N, j \in N} e_{ij} \tag{A.19}$$

where e_{ij} is the energy required to fly over edge ij. In addition to the nodes visited, a plan π_k defines the flight condition (airspeed, power and heading) along each edge.

The optimal plan π^* minimizes the energy required to fly the course:

$$E^*(\pi^*) = \min_\pi E(\pi) \tag{A.20}$$

This is computed using wavefront expansion from the start through each of the turnpoints until the goal is reached. The resulting plan π^* encodes the set of nodes visited (i.e. the optimal altitudes h_p^* for each turnpoint), the set of optimal airspeeds v_a^* and groundspeeds v_g^* for each segment, the set of optimal power P^* for each segment as well as the expected energy consumption for each segment.

This segment by segment optimal plan is not the global minimum energy trajectory: for example it does not include the capability to trade a higher speed descent at the end of the flight for a slower (possibly more efficient) climb at the beginning of the flight. Rather, the

segment by segment plan determines when it is advantageous to change altitude to exploit favorable wind (or avoid unfavorable wind).

A.6.2 Trajectory refinement

To refine the flight plan the problem of minimizing energy consumption is cast as a parameter optimization problem, with airspeed and power on each segment defining the parameters. The energy required to fly a segment is the energy required per unit distance ("gallons per mile") for that segment multiplied by segment length. The total plan cost is the sum of energies for each segment:

$$E = \sum_{i=1}^{N} \frac{P_i}{v_{gi}} \Delta s_i \tag{A.21}$$

The refined trajectory is computed by solving

$$\text{minimize } E \tag{A.22}$$

$$\text{subject to } C_{Pi} = f_P(J_i) \tag{A.23}$$

$$C_{Ti} = f_T(Ji) \tag{A.24}$$

$$0.01 \leq J_i \leq 1.5 \tag{A.25}$$

$$\gamma_{gi} = \gamma_{gi}^{req} \tag{A.26}$$

$$h_p^* - \Delta h \leq h_p \leq h_p^* + \Delta h \tag{A.27}$$

$$\sum_{i=1}^{N} \frac{\Delta s_i}{v_{gi}} \leq \frac{1}{v_{g,min}} \sum_{i=1}^{N} \Delta s_i \tag{A.28}$$

where h_p^* is the desired altitude at the destination turnpoint of the segment (from the segment-by-segment optimal trajectory) and Δh is a maximum allowable perturbation in altitude at a turnpoint to permit the path to find the local optimum. As in the segment flight condition optimization, the polynomial functions f_P and f_T define power and thrust coefficient in terms of advance ratio, respectively; γ_g defines the flight path with respect to the ground; finally the last constraint defines the minimum average ground speed.

This problem can be solved using a gradient descent optimizer (in this case MatLab's `fmincon`). The initial guess is the set of segment speeds v^* and power on each segment P^* that minimized the segment-by-segment plan.

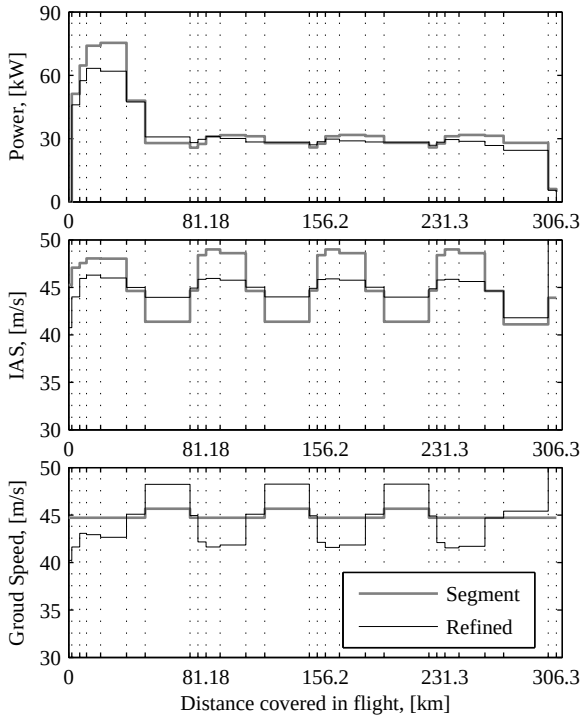


Figure A.7. Power Setting, Indicated Airspeed, and Ground Speed for the energy flight. The grey line shows the flight condition after the segment by segment optimization, the black line line shows flight condition after trajectory refinement.

An example trajectory refinement is shown in Figure A.7 using wind field data from Sept 27, 2011 (the day of the fuel efficiency flight). A vector plot of the wind field is shown in Figure A.12(c).

Note the change in ground speeds on each segment after refinement: a high ground speed during the descent segments is traded for lower ground speed during climb. Also, ground speeds change as the aircraft goes from tailwind segments to headwind segments.

The segment-by-segment optimized trajectory requires 67 kWh (equivalent to 1.99 gallons automotive gasoline); refining the flight plan reduces required energy by 1% to 66.4 kWh (equivalent to 1.97 gallons). Figure A.8 shows energy consumption with distance travelled over the course for both the segment by segment plan and the refined plan.

The speed flight minimization followed a similar routine as the energy flight, but time required to fly the course was now minimized rather than energy required. An additional constraint of reserve energy was also implemented. This allowed the aircraft to fly as fast as possible while maintaining enough energy to satisfy the 30 minutes reserve energy requirement.

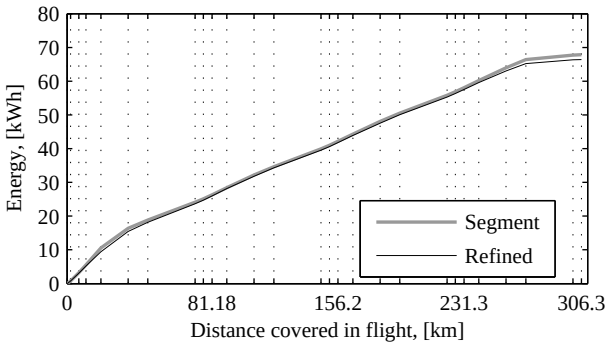


Figure A.8. Predicted minimum required energy consumption for the energy flight. The grey line shows energy consumption after the segment by segment optimization, the black line line shows energy consumption after trajectory refinement.

$$\text{minimize} \sum_{i=1}^{N} \frac{\Delta s_i}{v_{gi}} \tag{A.29}$$

$$\text{subject to } C_{Pi} = f_P(J_i) \tag{A.30}$$

$$C_{Ti} = f_T(Ji) \tag{A.31}$$

$$0.01 \leq J_i \leq 1.5 \tag{A.32}$$

$$\gamma_{gi} = \gamma_{gi}^{req} \tag{A.33}$$

$$h_p^* - \Delta h \leq h_p \leq h_p^* + \Delta h \tag{A.34}$$

$$E \leq E_T - E_{rsv} \tag{A.35}$$

where E is energy consumed, E_T is initial energy available in the batteries and E_{rsv} is the required reserve at the end of the flight.

This was initialized with a guess based on the predicted flight path of the energy flight. The wind speed gradient at each turnpoint was again assumed negligible to allow marginal variation in altitude. The results from these refined minimizations were used in hard-copy flight plans for the pilots as a means of guidelines. The flight path could obviously change due to unpredictable changes in the weather during the time of the competition. It was then at the pilot's discretion to fly as close to the predicted flight plan as warranted. This trajectory optimization was proven to be beneficial by comparing the initial and final values of the minimization routine in figure A.8.

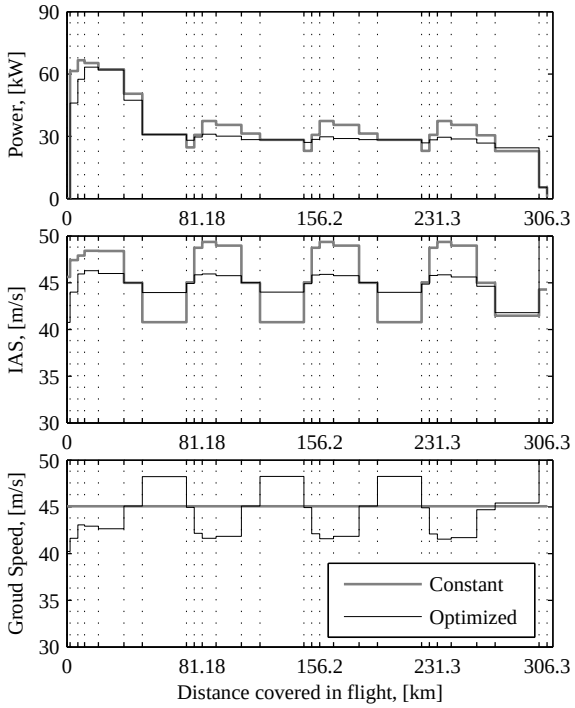


Figure A.9. Comparison of energy-optimized flight plan with constant ground speed flight plan.

A.6.3 The utility of flight planning

Competition flights were intentionally scheduled for mornings, when prevailing winds are typically calm in the Santa Rosa area. Further, the "tight" course reduces the utility of altitude changes to seek favorable winds. The segment by segment optimal trajectory thus seeks the minimum altitude path around the course while meeting the minimum ground speed constraint.

However, over a closed path such as the GFC course winds will always have an adverse effect, and even for this case an optimal flight plan is better than alternatives. Consider for example a flight plan that simply flies constant ground speed (e.g. 100 mph) at the lowest possible altitude, thus satisfying the ground speed constraint. The energy cost of a constant ground speed flight plan is 67.0 kWh versus 66.4 kWh for the refined flight plan.

A.7 Competition Flights

The CAFE Foundation provided a power and total consumed energy measuring device (eTotalizer) and a GPS receiver to track position and ground speed.

A.7.1 Efficiency flight: September 27, 2011

A.7.1.1 Atmospheric conditions

The competition day was mostly clear weather. By 1000PDT the temperature at STS was approximately 20°C, pressure was 30.02 inHg and surface winds were calm. Because of a temperature inversion (which persisted throughout the morning) temperature at 4000' MSL was about 15°C above standard, giving a density altitude of 6000'. Figure A.12 shows winds at 4000' MSL over the course: initial flow from the north east shifted to flow from the north as the day progressed, with terrain influencing flow direction over some parts of the course.

A.7.1.2 Planned and actual flight conditions

Take-off occurred at about 1000 PDT and the flight plan using the wind field for 1100PDT was used for the flight (which was expected to last under two hours).

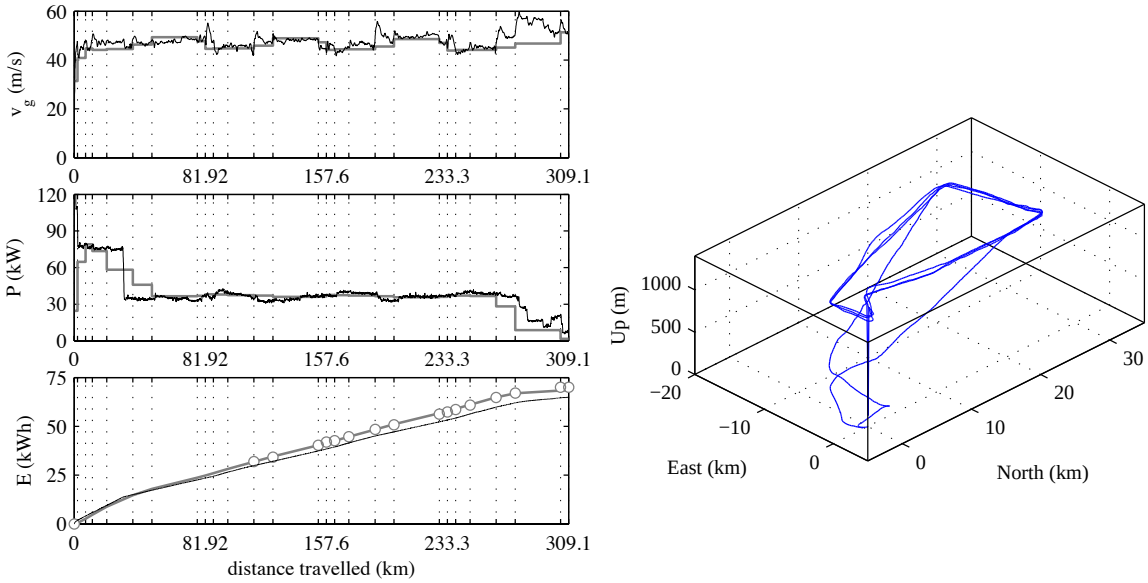
A comparison of planned ground speed, power and energy consumption with actual data is shown in Figure A.10. Tick marks along the x axis are placed at turnpoints: total length of the planned flight was scaled to match the actual flight distance from the GPS trace. Figure A.10(b) shows the flight path.

There is a difference between the planned energy consumption and the eTotalizer, while pilot reported consumption matches planned consumption within 1%. The pilot energy consumption display obtains its data from a different source than the CAFE eTotalizer, and there is a difference in calibration.

A.7.2 Speed flight: September 29, 2011

A.7.2.1 Atmospheric conditions

All aircraft took off by about 0930 PDT. Surface temperature at 1000 PDT was approximately 21°C, pressure was 29.90 inHg and surface winds were from the south at 3.5 mph. By noon PDT temperature had increased to 25°C and surface winds had increased to 6.9 mph. The temperature inversion was again present, giving a density altitude of 6000' at 4000' geometric altitude. Figure A.13 shows winds at 4000' MSL over the course: initial flow from the south shifted to flow from the south east as the day progressed, with magnitude increasing somewhat.



(a) Flight data

(b) GPS trace

Figure A.10. Flight data and trace of energy flight. Left: Ground speed, power and energy consumption for efficiency flight. Solid black line shows data from eTotalizer, grey line is planned flight condition, open circles show pilot observed energy consumption; Right: trace of GPS positions.

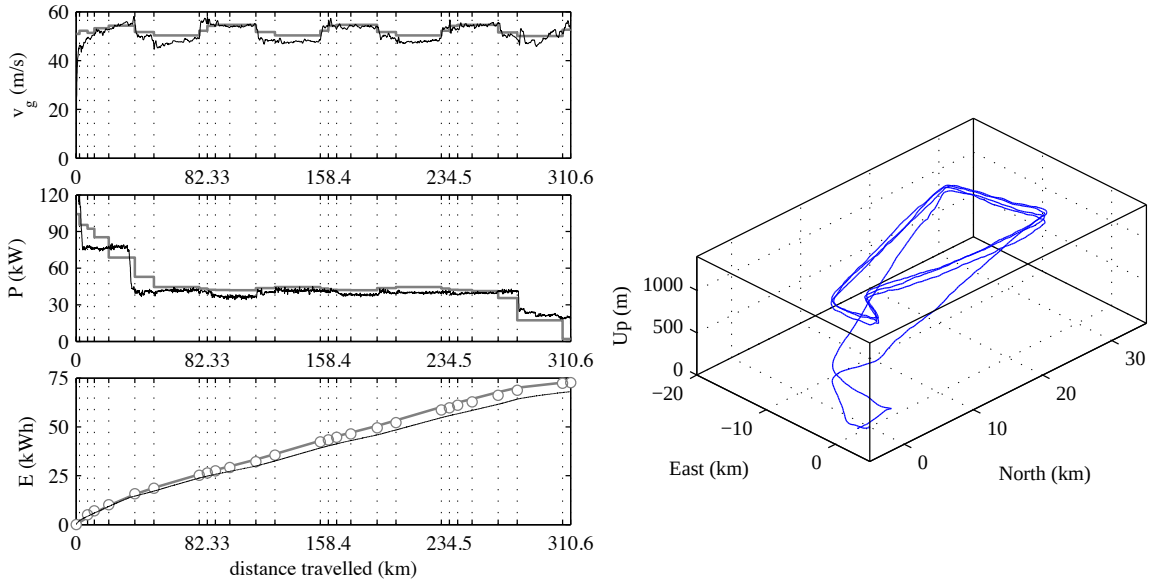
A.7.2.2 Planned and actual flight conditions

With takeoff occurring at 0930PDT wind conditions for 1000PDT were used for the flight plan. The goal of the flight was to fly as fast as possible and arrive at the finish having consumed 72 kWh (on the cockpit display of energy consumption). This would give about 2 kWh leeway for the required reserve energy available.

Flight data is shown in Figure A.11(a) and a GPS trace is shown in Figure A.11(b). Again the planned distance has been scaled to match the distance flown, and the same calibration difference is visible between the official eTotalizer data and the cockpit displayed data.

A.7.3 Competition results

Final results obtained from the competition is given in Table A.3. The total energy consumed was within 1% of what was predicted through simulations.We won the Green Flight Challenge. Meteorological wind preconditions and flight path planning beforehand helped to achieve this incredible feat.



(a) Flight data (b) GPS trace

Figure A.11. Flight data and trace of speed flight. Left: Ground speed, power and energy consumption for speed flight. Solid black line shows data from eTotalizer, grey line is planned flight condition, open circles show pilot observed energy consumption; Right: trace of GPS positions.

Table A.3. Final results of 2011 Green Flight Challenge[6]. Energy consumption and speeds are from CAFE Foundation measuring equipment.

	Pipistrel-USA.com	**units**
Efficiency flight		
electricity	65.4	kWh
equivalent fuel	1.94 (7.34)	gal (L) mogas
flight time	1:49:37	h:mm:ss
distance	195.9 (315.3)	miles (km)
speed	107.4 (172.8)	mph (km/h)
mileage	403.5	pMPGe
Speed flight		
electricity	68.3	kWh
equivalent fuel	2.03 (7.68)	gal (L) mogas
flight time	1:41:55	h:mm:ss
distance	193.0 (310.6)	miles (km)
speed	113.6 (182.8)	mph (km/h)
mileage	388.4	pMPGe

A.8 Pros and Cons of Graph Based Planning

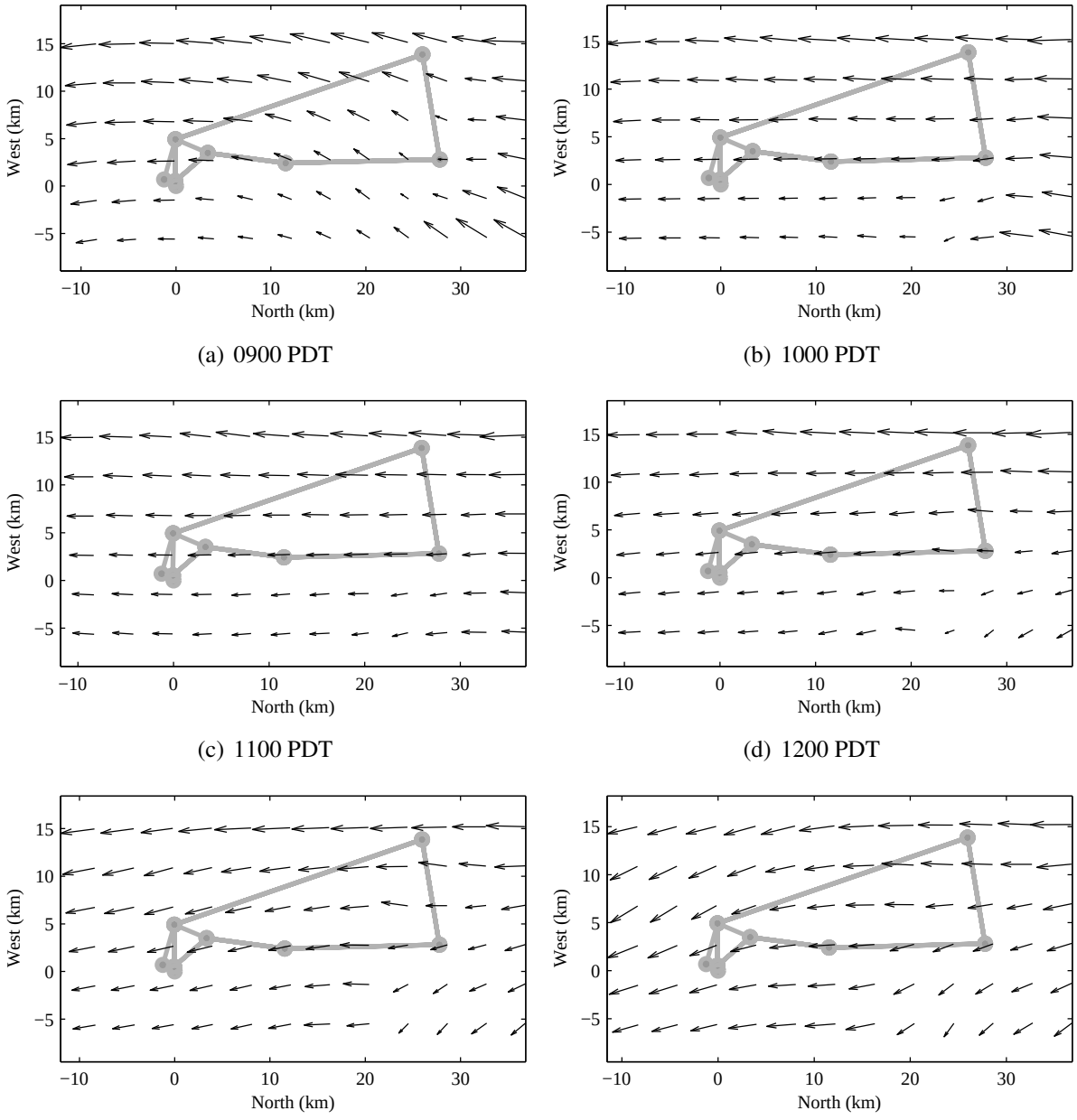
Graph based planning technique was very successful in the Green Flight Challenge. It allowed for segment by segment optimization and over all optimization of the path. All this was possible because a fixed path was given to follow and time of launch of flight was also specified. The restriction of always flying within three miles of the specified path ensured that our only allowance was that we can change altitude if needed. This essentially reduced the search space in a 2-D grid. Also ground velocity is always positive, thus always moving towards the goal constraint was satisfied. Thus the energy-map approach was best suited for the Green Flight Challenge.

The general problem of path planning in 4-D (three spatial dimensions and time)is much more complex. Moving from 2-D to 3-D planning increases the allowable transitions over three times. Moreover considering time dimension makes the search space significantly large.

Winds vary significantly during a day. For the Green Flight Challenge planning at different starting times of the day was done in advance. To plan in a time varying wind field the planner should provide for a precise starting time which will be most beneficial from the energy point of view. Sampling based motion planning is explored which explicitly accounts for three spatial dimensions and time.

A.9 Summary

This section has demonstrated a practical application of flight planning technique applied for a real experiment. In this section graph based planning technique was discussed in the context of Green Flight Challenge. The optimization problem was discussed for flight between two specified points in space. How optimized flight between nodes can be incorporated into graph based planning is discussed. Flight path planning and competition results are discussed in details. Finally limitations of graph based planning are summarized.



(a) 0900 PDT

(b) 1000 PDT

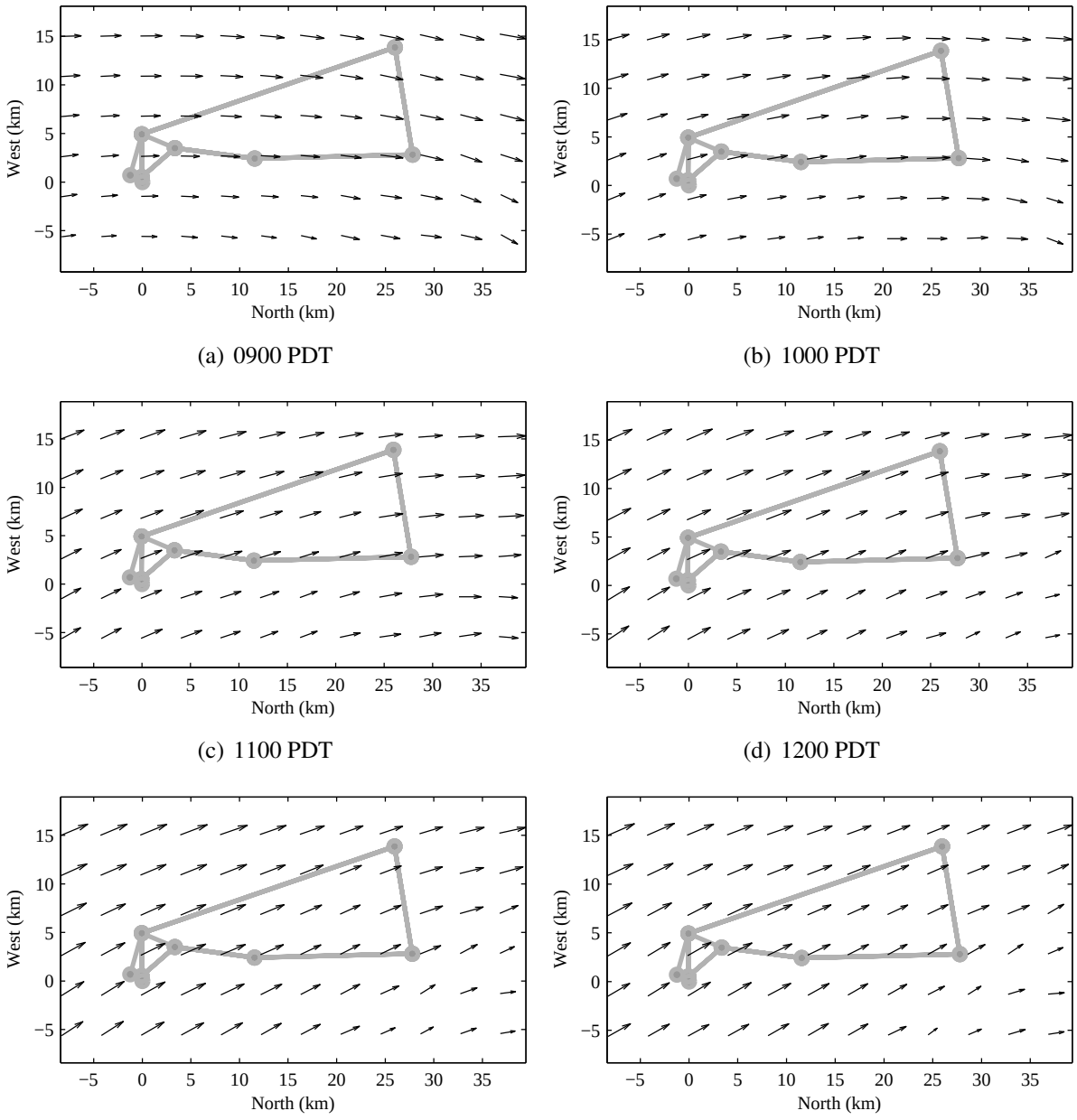
(c) 1100 PDT

(d) 1200 PDT

(e) 1300 PDT

(f) 1400 PDT

Figure A.12. Predicted winds over the course at 4000 feet MSL for September 27, 2011 (efficiency flight).



(a) 0900 PDT

(b) 1000 PDT

(c) 1100 PDT

(d) 1200 PDT

(e) 1300 PDT

(f) 1400 PDT

Figure A.13. Winds over the course at 4000 feet MSL for September 29, 2011 (speed flight).

Appendix B

Vehicle Properties of SB-XC

The simulations used a RNR glider called SB-XC. Figure B.1 shows a SB-XC glider and Table B.1 tabulates the properties of the glider.

Note that a fourth order polynomial is used to relate C_D to C_L: this provided a better fit to the computed data over the full speed range.

Figure B.1. SB-XC Glider. Image source: http://www.rcgroups.com/forums/.

Table B.1. Parameters for SB-XC glider.

variable	value	description
m	10 kg	mass
S	1 m^2	wing area
$f(C_L)$	$0.1723C_L^4 - 0.3161C_L^3 + 0.2397C_L^2$ $-0.0624C_L + 0.0194$	
$v_{a,min}$	12 m/s	
$v_{a,max}$	35 m/s	

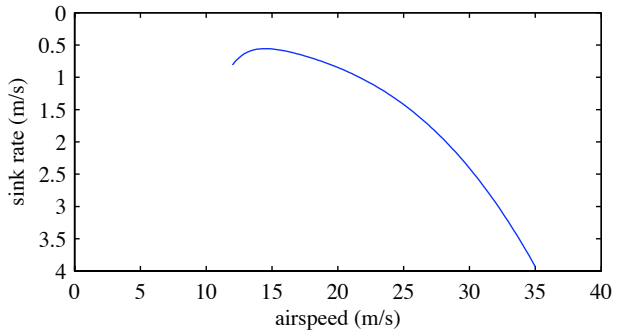


Figure B.2. Sink rate vs. airspeed for the SB-XC. Minimum sink is approximately 0.56 m/s and occurs at approximately 14.6 m/s.

B.1 The Nimbus III DM

B.1.1 Calculation of drag polar for Nimbus

The Nimbus III DM is a high performance single-seater glider.

Table B.2. Parameters for Nimbus III DM.

variable	value	description
m	820 kg	mass
S	16.85 m^2	wing area
p	24.6 m	wing span
$v_{a,min}$	30 m/s	
$v_{a,max}$	65 m/s	
$L/D\|_{max}$	57	best glide ratio

From the operating manual of the Nimbus III DM the drag polar can be calculated. The operating manual of the drag polar gives sink rates at different speed which is summarized in Table B.3

Figure B.3. The Nimbus III-DT glider.

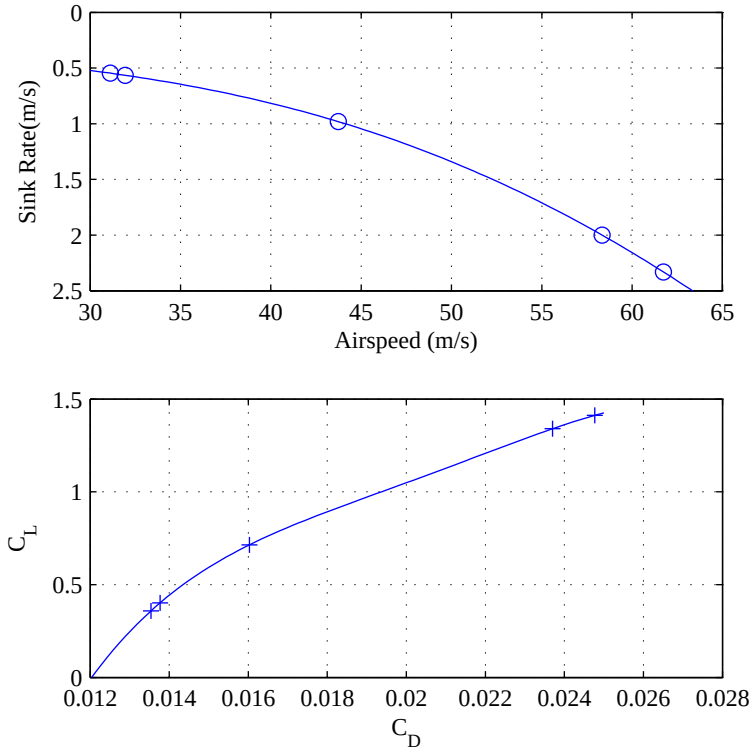


Figure B.4. The Nimbus III-DM polar plots.

Table B.3. Speed Polar for Nimbus III DM .

Speed Km/hr	**Sink Rate** m/s
114.97	-0.565
157.42	-0.981
210	-2.000
222.24	-2.331

Bibliography

[1] BOHRER, G., D. BRANDES, J. T. MANDEL, K. L. BILDSTEIN, T. A. MILLER, M. LANZONE, T. KATZNER, C. MAISONNEUVE, and J. A. TREMBLAY (2012) "Estimating updraft velocity components over large spatial scales: contrasting migration strategies of golden eagles and turkey vultures," *Ecology Letters*, **15**(2), pp. 96–103.

[2] CHAKRABARTY, A. and J. W. LANGELAAN (2011) "Energy-based Long-range Path Planning for Soaring-capable UAVs," *Journal of Guidance, Control and Dynamics*, **34**(4), pp. 1002–1015.

[3] LAVALLE, S. (2006) *Planning algorithms*, Cambridge Univ Pr.

[4] KARAMAN, S. and E. FRAZZOLI (2010) "Incremental sampling-based algorithms for optimal motion planning," in *Proc. Robotics: Science and Systems*.

[5] TOMAZIC, T., V. PLEVNIK, G. VEBLE, J. TOMAZIC, F. POPIT, S. KOLAR, R. KIELJ, J. W. LANGELAAN, and K. MILES (2011) "Pipistrel Taurus G4: on Creation and Evolution of the Winning Aeroplane of the NASA Green Flight Challenge 2011," *Strojniski vestnik - Journal of Mechanical Engineering*, **57**(12), pp. 869–878.

[6] HTTP://CAFEFOUNDATION.ORG/V2/GFC_2011_RESULTS.HTML *checked June 27, 2012*.

[7] RAYLEIGH, J. W. S. (1883) "The Soaring of Birds," *Nature*, **27**, pp. 534–535.

[8] ——— (1889) "The Sailing Flight of the Albatross," *Nature*, **40**, p. 34.

[9] BOSLOUGH, M. B. E. (2002) *Autonomous Dynamic Soaring Platform for Distributed Mobile Sensor Arrays*, *Tech. Rep. SAND2002-1896*, Sandia National Laboratories, Sandia National Laboratories.

[10] KICENIUK, T. (2001) "Calculations on Soaring in Sink," *Technical Soaring*, **25**(4), pp. 228–230.

[11] PATEL, C. K. and I. KROO (2006) "Control Law Design for Improving UAV Performance using Wind Turbulence," in *AIAA Aerospace Sciences Meeting and Exhibit*, AIAA Paper 2006-0231, American Institute of Aeronautics and Astronautics, Reno, Nevada.

[12] MICHALAKES, J., J. DUDHIA, D. GILL, T. HENDERSON, J. KLEMP, W. SKAMAROCK, and W. WANG (2004) "The weather research and forecast model: software architecture and performance," in *Proceedings of the 11th ECMWF Workshop on the Use of High Performance Computing In Meteorology*, vol. 25, World Scientific, p. 29.

[13] MICHALAKES, J., S. CHEN, J. DUDHIA, L. HART, J. KLEMP, J. MIDDLECOFF, and W. SKAMAROCK (2001) "Development of a next generation regional weather research and forecast model," in *Developments in Teracomputing: Proceedings of the Ninth ECMWF Workshop on the use of high performance computing in meteorology*, vol. 1, World Scientific, pp. 269–276.

[14] FROUDE, R. (1889) "Sailing flight of the Albatross," *Nature*, **40**, p. 102.

[15] SACHS, G. (2005) "Minimum shear wind strength required for dynamic soaring of albatrosses," *Ibis*, **147**(1), pp. 1–10.

[16] PENNYCUICK, C. (1960) "Gliding flight of the fulmar petrel," *Journal of experimental Biology*, **37**(2), pp. 330–338.

[17] ZERMELO, E. (1931) "Über das Navigationsproblem bei ruhender oder veränderlicher Windverteilung," *ZAMM-Journal of Applied Mathematics and Mechanics/Zeitschrift für Angewandte Mathematik und Mechanik*, **11**(2), pp. 114–124.

[18] HOTA, S. and D. GHOSE (2010) "Optimal path planning for an aerial vehicle in 3D space," in *Decision and Control (CDC), 2010 49th IEEE Conference on*, IEEE, pp. 4902–4907.

[19] DUBINS, L. E. (1957) "On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents," *American Journal of Mathematics*, **79**, pp. 497–516.

[20] SUSSMANN, H. J. (1997) "The Markov-Dubins problem with angular acceleration control," in *In Proccedings of the 36th IEEE Conference on Decision and Control*, IEEE Publications, pp. 2639–2643.

[21] MCGEE, T. and J. HEDRICK (2007) "Optimal path planning with a kinematic airplane model," *Journal of guidance, control, and dynamics*, **30**(2), pp. 629–633.

[22] MCGEE, T., S. SPRY, and J. HEDRICK (2005) "Optimal path planning in a constant wind with a bounded turning rate," in *AIAA Guidance, Navigation, and Control Conference and Exhibit*, Citeseer, pp. 1–11.

[23] BAKOLAS, E. and P. TSIOTRAS (2010) "Time-Optimal Synthesis for the Zermelo-Markov-Dubins Problem: the Constant Wind Case," in *Proceedings of the American Control Conference*, Baltimore, Maryland.

[24] RAO, D. and S. B. WILLIAMS (2009) "Large-scale path planning for Underwater Gliders in ocean currents," in *Australasian Conference on Robotics and Automation (ACRA), Sydney*.

[25] GARAU, B., M. BONET, A. ALVAREZ, S. RUIZ, A. PASCUAL, ET AL. (2009) "Path planning for autonomous underwater vehicles in realistic oceanic current fields: Application to gliders in the western mediterranean sea," *Journal of Maritime Research*, **6**(2), pp. 5–22.

[26] LOLLA, T., M. UECKERMANN, K. YIGIT, P. HALEY JR, and P. F. LERMUSIAUX (2012) "Path planning in time dependent flow fields using level set methods," in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, IEEE, pp. 166–173.

[27] KAHVECI, N., P. IOANNOU, and D. MIRMIRANI (2007) "Optimal static soaring of uavs using vehicle routing with time windows," in *AIAA Aerospace Sciences Meeting and Exhibit, AIAA-2007-158, AIAA, Washington, DC, USA*.

[28] GORDON, R. J. (2006) *Optimal dynamic soaring for full size sailplanes*, *Tech. rep.*, DTIC Document.

[29] ALLEN, M. J. (2005) "Autonomous Soaring for Improved Endurance of a Small Uninhabited Air Vehicle," in *43rd AIAA Aerospace Sciences Meeting and Exhibit*, American Institute of Aeronautics and Astronautics, Reno, Nevada.

[30] ALLEN, M. J. and V. LIN (2007) "Guidance and Control of an Autonomous Soaring Vehicle with Flight Test Results," in *AIAA Aerospace Sciences Meeting and Exhibit*, AIAA Paper 2007-867, American Institute of Aeronautics and Astronautics, Reno, Nevada.
URL `http://dtrs.dfrc.nasa.gov/archive/00001275/`

[31] EDWARDS, D. J. (2008) "Implementation Details and Flight Test Results of an Autonomous Soaring Controller," in *AIAA Guidance, Navigation and Control Conference*, American Institute of Aeronautics and Astronautics, Reston, Virginia.

[32] TORROELLA, J. C. R. (2004) *Long Range Evolution-based Path Planning for UAVs through Realistic Weather Environments*, Master's thesis, University of Washington, Seattle, Washington.
URL `www.aa.washington.edu/research/afsl/publications/rubio2004thesis.pd`

[33] JARDIN, M. R. and A. E. BRYSON (2001) "Neighboring Optimal Aircraft Guidance in Winds," *Journal of Guidance, Control and Dynamics*, **24**(4), pp. 710–715.

[34] MACCREADY JR., P. B. (1958) "Optimum Airspeed Selector," *Soaring*, pp. 10–11.

[35] COCHRANE, J. H. (1999) "MacCready Theory with Uncertain Lift and Limited Altitude," *Technical Soaring*, **23**(3), pp. 88–96.
URL http://faculty.chicagogsb.edu/john.cochrane/research/Papers/newmcre

[36] REICHMANN, H. (1978) *Cross-Country Soaring*, Thomson Publications, Santa Monica, California.

[37] ARHO, R. (1974) "Optimal Dolphin Soaring as a Variational Problem," in *OSTIV Publication XIII*, Organisation Scientifique et Technique Internationale du Vol à Voile.

[38] METZGER, D. E. and J. K. HEDRICK (1975) "Optimal Flight Paths for Soaring Flight," *Journal of Aircraft*, **12**(11), pp. 867–871.

[39] SANDAUER, J. (1978) "Some Problems of the Dolphin-Mode Flight Technique," in *OSTIV Publication XV*, Organisation Scientifique et Technique Internationale du Vol à Voile.

[40] DE JONG, J. L. (1981) "The Convex Combination Approach: A Geometric Approach to the Optimization of Sailplane Trajectories," in *OSTIV Publication XVI*, Organisation Scientifique et Technique Internationale du Vol à Voile, pp. 182–201.

[41] PIERSON, B. L. and I. CHEN (1979) "Minimum Altitude Loss Soaring in a Specified Vertical Wind Distribution," in *NASA Conference Publication 2085, Science and Technology of Low Speed and Motorless Flight* (P. W. Hanson, ed.), NASA, Hampton, Virginia, pp. 305–318.

[42] SANDER, G. and F. X. LITT (1979) "On Global Optimal Sailplane Flight Strategy," in *NASA Conference Publication 2085, Science and Technology of Low Speed and Motorless Flight* (P. W. Hanson, ed.), NASA, Hampton, Virginia, pp. 355–376.

[43] LOZANO-PEREZ, T. (1983) "Spatial planning: A configuration space approach," *IEEE transactions on computers*, **100**(32), pp. 108–120.

[44] LOZANO-PÉREZ, T. and M. WESLEY (1979) "An algorithm for planning collision-free paths among polyhedral obstacles," *Communications of the ACM*, **22**(10), p. 570.

[45] STENTZ, A. (1995) "The Focussed D* Algorithm for Real-Time Replanning," in *Proceedings of the International Joint Conference on Artificial Intelligence*.

[46] ANISI, D. A., J. W. C. ROBINSON, and P. OGREN (2006) "On-line Trajectory Planning for Aerial Vehicles: a Safe Approach with Guaranteed Task Completion," in *AIAA Guidance, Navigation and Control Conference*, AIAA Paper 2006-6107, American Institute of Aeronautics and Astronautics, Keystone, Colorado.

[47] DIJKSTRA, E. W. (1959) "A note on two problems in connexion with graphs," *Numerische mathematik*, **1**(1), pp. 269–271.

[48] HART, P. E., N. J. NILSSON, and B. RAPHAEL (1968) "A formal basis for the heuristic determination of minimum cost paths," *Systems Science and Cybernetics, IEEE Transactions on*, **4**(2), pp. 100–107.

[49] NILSSON, N. J. (1982) *Principles of artificial intelligence*, Springer.

[50] BELLMAN, R. (1956) "Dynamic programming and Lagrange multipliers," *Proceedings of the National Academy of Sciences of the United States of America*, **42**(10), p. 767.

[51] CORMEN, T. H., C. E. LEISERSON, R. L. RIVEST, and C. STEIN (2001) *Introduction to Algorithms*, second edition ed., MIT Press, Cambridge, Massachusetts.

[52] BARBEHENN, M. and S. HUTCHINSON (1995) "Efficient search and hierarchical motion planning by dynamically maintaining single-source shortest paths trees," *Robotics and Automation, IEEE Transactions on*, **11**(2), pp. 198–214.

[53] KOENIG, S. and M. LIKHACHEV (2002) "Improved fast replanning for robot navigation in unknown terrain," in *Robotics and Automation, 2002. Proceedings. ICRA'02. IEEE International Conference on*, vol. 1, IEEE, pp. 968–975.

[54] STENTZ, A. (1995) "The focussed Dˆ* algorithm for real-time replanning," in *IJCAI*, vol. 95, pp. 1652–1659.

[55] FERGUSON, D., M. LIKHACHEV, and A. STENTZ (2005) "A guide to heuristic-based path planning," in *Proceedings of the international workshop on planning under uncertainty for autonomous systems, international conference on automated planning and scheduling (ICAPS)*, pp. 9–18.

[56] CHAKRABARTY, A. and J. W. LANGELAAN (2010) "Flight path planning for uav atmospheric energy harvesting using heuristic search," in *AIAA Guidance, Navigation and Controls Conference, Toronto, Canada*.

[57] REIF, J. (1979) "Complexity of the mover's problem and generalizations extended abstract," in *Proceedings of the 20th Annual IEEE Conference on Foundations of Computer Science*, pp. 421–427.

[58] KONDO, K. (1991) "Motion planning with six degrees of freedom by multistrategic bidirectional heuristic free-space enumeration," *Robotics and Automation, IEEE Transactions on*, **7**(3), pp. 267–277.

[59] BARRAQUAND, J. and J.-C. LATOMBE (1990) "A Monte-Carlo algorithm for path planning with many degrees of freedom," in *Robotics and Automation, 1990. Proceedings., 1990 IEEE International Conference on*, IEEE, pp. 1712–1717.

[60] GLAVINA, B. (1990) "Solving findpath by combination of goal-directed and randomized search," in *IEEE Int. Conf. Robot. & Autom*, pp. 1718–1723.

[61] KAVRAKI, L., P. SVESTKA, J. LATOMBE, and M. OVERMARS (1996) "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *Robotics and Automation, IEEE Transactions on*, **12**(4), pp. 566–580.

[62] LAVALLE, S. and J. KUFFNER JR (2001) "Randomized kinodynamic planning," *The International Journal of Robotics Research*, **20**(5), pp. 378–400.

[63] JAILLET, L., J. CORTÉS, and T. SIMÉON (2010) "Sampling-based path planning on configuration-space costmaps," *Robotics, IEEE Transactions on*, **26**(4), pp. 635–646.

[64] CHENG, P. and S. LAVALLE (2001) "Reducing metric sensitivity in randomized trajectory design," in *Intelligent Robots and Systems, 2001. Proceedings. 2001 IEEE/RSJ International Conference on*, vol. 1, IEEE, pp. 43–48.

[65] ——— (2002) "Resolution complete rapidly-exploring random trees," in *Robotics and Automation, 2002. Proceedings. ICRA'02. IEEE International Conference on*, vol. 1, IEEE, pp. 267–272.

[66] FRAZZOLI, E., M. A. DAHLEH, and E. FERON (1999) "A hybrid control architecture for aggressive maneuvering of autonomous helicopters," in *Decision and Control, 1999. Proceedings of the 38th IEEE Conference on*, vol. 3, IEEE, pp. 2471–2476.

[67] METTLER, B., M. VALENTI, T. SCHOUWENAARS, E. FRAZZOLI, and E. FERON (2002) "Rotorcraft motion planning for agile maneuvering," *Montreal, QC, June*.

[68] PEARL, J. (1984) "Heuristics: intelligent search strategies for computer problem solving," .

[69] HUYN, N., R. DECHTER, and J. PEARL (1980) "Probabilistic analysis of the complexity of A," *Artificial Intelligence*, **15**(3), pp. 241–254.

[70] HARRIS, T. E. (2002) *The theory of branching processes*, Courier Dover Publications.

[71] POHL, I. (1969) *First results on the effect of error in heuristic search*, Edinburgh University, Department of Machine Intelligence and Perception.

[72] PEARL, J. (1981) "Heuristic Search Theory: Survey of Recent Results." in *IJCAI*, vol. 1, pp. 554–562.

[73] SKAMAROCK, W. C., J. B. KLEMP, J. DUDHIA, D. O. GILL, D. M. BARKER, M. G. DUDA, X. HUANG, W. WANG, and J. G. POWERS (2008) *A description of the Advanced Research WRF Version 3*, *Tech. Rep. NCAR/TN-475-STR*, National Center for Atmospheric Research.

[74] GEORGE YOUNG, N. L. S., B. GAUDET and D. R. STAUFFER (2009) "Interaction of a mountain lee wave with a basin cold pool," in *13th Conference on Mesoscale Processes*, AMS, American Meteriological Society, Chicago, Illinois.

[75] HTTP://WWW.CLOUDCAPTECH.COM/PICCOLOSYSTEM.SHTM .

[76] LUMELSKY, V. J. and A. A. STEPANOV (1986) "Dynamic path planning for a mobile automaton with limited information on the environment," *Automatic Control, IEEE Transactions on*, **31**(11), pp. 1058–1063.

[77] QUINDLEN, J. and J. LANGELAAN (2013) "Flush Air Data Sensing for Soaring-Capable UAVs," in *51st AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition*.

[78] FREW, E., J. ELSTON, B. ARGROW, A. HOUSTON, and E. RASMUSSEN (2012) "Sampling Severe Local Storms and Related Phenomena: Using Unmanned Aircraft Systems," *Robotics Automation Magazine, IEEE*, **19**(1), pp. 85–95.

[79] BROWN, S. T., B. LAMBRIGTSEN, R. F. DENNING, T. GAIER, P. KANGASLAHTI, B. H. LIM, J. M. TANABE, and A. B. TANNER (2011) "The high-altitude MMIC sounding radiometer for the global hawk unmanned aerial vehicle: Instrument description and performance," *Geoscience and Remote Sensing, IEEE Transactions on*, **49**(9), pp. 3291–3301.

[80] LIPINSKI, D. and K. MOHSENI "Feasible area coverage of a hurricane using micro-aerial vehicles," .

[81] ELSTON, J. S., J. ROADMAN, M. STACHURA, B. ARGROW, A. HOUSTON, and E. FREW (2011) "The tempest unmanned aircraft system for in situ observations of tornadic supercells: design and VORTEX2 flight results," *Journal of Field Robotics*, **28**(4), pp. 461–483.

[82] CHAKRABARTY, A. and J. LANGELAAN (2013) "UAV flight path planning in time varying complex wind-fields," in *American Control Conference (ACC), 2013*, IEEE, pp. 2568–2574.

[83] SLOTNICK, J., R. W. CLARK, D. M. FRIEDMAN, Y. YADLIN, D. T. YEH, J. E. CARR, M. J. CZECH, and S. W. BIENIAWSKI (2014) "Computational Aerodynamic Analysis for the Formation Flight for Aerodynamic Benefit Program," in *AIAA Science and Technology Forum and Exposition*.

[84] MCCORMICK, B. W. (1995) *Aerodynamics, Aeronautics and Flight Mechanics*, second ed., John Wiley & Sons, Inc., Hoboken, NJ.

[85] RUBIO, J. C. and S. KRAGELUND (2003) "The trans-pacific crossing: long range adaptive path planning for UAVs through variable wind fields," in *Proceedings of the 22nd Digital Avionics Systems Conference*, vol. 2, IEEE, Piscataway, New Jersey.

[86] KUWATA, Y. and J. P. HOW (2004) "Three Dimensional Receding Horizon Control for UAVs," in *AIAA Guidance, Navigation and Control Conference*, AIAA Paper 2004-5144, American Institute of Aeronautics and Astronautics, Reston, Virginia.

[87] SCHOUWENAARS, T., B. METTLER, E. FERON, and J. P. HOW (2004) "Hybrid Model for Trajectory Planning of Agile Autonomous Vehicles," *Journal of Aerospace Computing, Information and Communication*, **1**, pp. 629–651.

Vita

Anjan Chakrabarty

Education

The Pennsylvania State University, University Park, PA, USA
Ph.D Candidate in Aerospace Engineering, August 2014
Advisor: Dr. Jack Langelaan
Research : Energy Aware Path Planning in Complex Four Dimensional Environments.
MS in Aerospace Engineering, July 2010
Advisor: Dr. Jack Langelaan
Thesis: Flight Path Planning of UAV Based on Atmospheric Energy Harvesting

Jadavpur University Kolkata, India
BE in Electrical Engineering, July 2007
Advisor: Prof. Tapan Kumar Ghoshal
Thesis: Modeling and Simulation of Atmospheric Reentry Trajectory applicable to reusable launch vehicle.

Research Experience

Graduate Research Assistant - Pennsylvania State University, Aerospace Engineering Department, Advisor- Dr. Jack Langelaan - August 2008-till date
Research Assistant- Indian Institute of Science, Aerospace Engineering Department, Advisor- Prof. M. Seetharama Bhat, August 2007 - June 2008
Summer Internship- Indian Institute of Science, Aerospace Engineering Department, Advisor- Dr. Radhakant Padhi, June- August, 2006

Awards

NASA Green Flight Challenge 2011- Team member of Winning team Pipistrel-USA.com. Our aircraft, the **Collier trophy nominated Taurus G4**, flew 403 passenger miles per gallon over a 195 mile course at 107 miles per hour.

First National Flying Competition MICAV 2007, Agra,India. Team member IISc. Won the Prize for Best Stabilized Flight. The competition was jointly organized by ADRDE(Aerial Delivery Research and Development Establishment) and NAL (National Aerospace Laboratories).

Best Presentation in Session Award, American Control Conference, Washington DC, 2013