

**A FRAMEWORK FOR OFFLINE RISK-AWARE PLANNING OF
LOW-ALTITUDE AERIAL FLIGHTS DURING URBAN DISASTER RESPONSE**

A Dissertation
Presented to
The Academic Faculty

By

Caleb M. Harris

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Computational Science and Engineering
Aerospace Engineering Home Unit

Georgia Institute of Technology

December 2022

© Caleb M. Harris 2022

A FRAMEWORK FOR OFFLINE RISK-AWARE PLANNING OF LOW-ALTITUDE AERIAL FLIGHTS DURING URBAN DISASTER RESPONSE

Thesis committee:

Dr. Dimitri Mavris, Advisor
Aerospace Engineering
Georgia Institute of Technology

Dr. Daniel Schrage
Aerospace Engineering
Georgia Institute of Technology

Dr. Polo Chau
Computational Science and Engineering
Georgia Institute of Technology

Dr. Youngjun Choi
Director of Advanced AI and Robotics
UPS ATG Robotics AI Lab (RAIL)

Dr. Richard Vuduc
Computational Science and Engineering
Georgia Institute of Technology

Date approved: October 3, 2022

If I'm an advocate for anything, it's to move. As far as you can, as much as you can.
Across the ocean, or simply across the river. Walk in someone else's shoes or at least eat
their food. It's a plus for everybody.

Anthony Bourdain

For my wife, Minami, and our dog, Leo.

ACKNOWLEDGMENTS

First, I want to thank all my committee members for their feedback and guidance during this work. I would like to thank Dr. Daniel Schrage for five years of mentorship through projects and courses that started in my first class at Georgia Tech, Graduate Rotorcraft Design. That class inspired me to continue into the exciting and unpredictable world of modern rotorcraft. I want to thank Dr. Polo Chau and Dr. Richard Vuduc, who have been critical to my success in the CSE program that started in their classes and continued through my CSE qualification exams and thesis. I want to thank Dr. Youngjun Choi, Director of Advanced AI and Robotics at UPS ATG Robotics AI Lab (RAIL), for dedicating time to this Thesis and my career. I want to thank you for all the fun memories with the DroneX team and the many hours of mentorship. I hope to continue working together in the future. I must give a special thanks to my advisor and head of the Aerospace Systems Design Lab, Dr. Dimitri Mavris. I sincerely appreciate all the support and confidence you have provided me, and I thank you for never giving up on me.

Thank you to Colin Bell at Brinc Drones for the insight into aerial system disaster response. I wish you and the Brinc team the best of luck in the important work you all are doing. In addition, thank you to Marco Sterk for some of my earliest mentorship at the University of Memphis and Express Drone Parts. You were critical to opening my imagination up for aerospace, autonomous systems, and innovation. I hope to stay in touch and continue working on the future of aerospace technology together.

I would also like to thank those that have provided support and guidance within the Aerospace Systems Design Lab. Thank you to Cedric Justin and Alexia Payan for mentoring me and putting their trust in me through multiple years of research projects. Thank you to my close friends from Weber office 315, Thomas Lin and Tim Chin. I learned a lot through our afternoon control chats and late-night group projects. Similar conversations helped me through the research and courses of graduate school, and I would like to thank

the friendship and collaboration of Etienne Demers Bouchard, Hyun Ki Lee, Seulki Kim, Gabriel Achour, and Johnie Sublett. If I forgot anyone else who I have worked closely with, I apologize, but blame it on the all the space the Thesis has took in my brain. Also, I would like to thank Adrienne Durham in the front office of ASDL for answering my endless questions, especially over the last two years as I have navigated complexities of the CSE program and the Thesis dissertation process.

Thank you to all my close friends and the members of my *party*, Jacob Stickney, Jack Casey, Jimmy Tang, Gene Chen, and Michael Bolt. Our many adventures brought me great joy and kept me going through tough times. I must give a special thanks to my friend, Alexander Parmley. I cannot thank you enough for the feedback, English reviews, beautiful graphic art from aparmley.com, and most importantly, the friendship along the way. I appreciate it more than you know and am so glad to call you a dear friend.

Last but not least, thank you to all of my family who has supported me and helped me stay healthy and happy through my graduate studies. My family has spent considerable energy supporting me without any conditions. Your love and support gave me the joy and focus I needed to finish this Thesis. A quick thanks to my dog, Leo, who may not be able to read but knows how to cheer me up when I am down. A special thanks to my wife, Minami. Your love and support were the most crucial factors for me completing this work. You took this journey with me, and I look forward to our future together on the other side.

TABLE OF CONTENTS

Acknowledgments	v
List of Tables	xii
List of Figures	xiv
List of Acronyms	xix
Summary	xxi
Chapter 1: Introduction to Aerial System Disaster Response	1
1.1 Disaster Events	1
1.1.1 Types of Major Disasters	2
1.1.2 The Cost of Disasters	3
1.2 Disaster Response and Rescue	4
1.2.1 Technological Advancements for Disaster Response	5
1.3 Aerial and Autonomous Systems	8
1.4 Research Gaps	14
1.4.1 Current State of the Art	15
1.4.2 Trajectory Planning	16
1.4.3 Decision-Making	21

1.4.4	Safe-Flight	21
1.4.5	Modeling and Simulation	22
1.5	A Framework for SAFER Response	23
Chapter 2: A Framework for Disaster Response with Unmanned Aerial Systems		24
2.1	Urban Map Modeling	24
2.2	Offline Aerial Flight Planning	25
2.3	Safe Flight Under Uncertainty	27
2.4	Recent Contributions to Research	28
2.5	SAFER Framework Contributions	29
Chapter 3: Rapid Urban Mapping with Limited Data		33
3.1	Introduction	33
3.2	Background	34
3.2.1	Urban Map Modeling	35
3.2.2	Geospatial Information Systems	36
3.3	Research Question 1	36
3.4	Urban Map Creation	39
3.4.1	Terrain Model	40
3.4.2	Structures	48
3.4.3	Weather	53
3.4.4	Summary and Results	54
3.5	Urban Map Updates	56
3.5.1	Learning-based Map Updates with Aerial Imagery	60

3.5.2	Approach to Urban Labeling	63
3.5.3	Network Performance	74
3.5.4	Results	77
3.6	Summary and Future Work	79
Chapter 4: Efficient Three-Dimensional Offline Kinodynamic Planning		82
4.1	Introduction	82
4.1.1	Algorithm Selection	86
4.2	Background	86
4.2.1	Rapidly Exploring Random Trees	89
4.2.2	Kinodynamic Planning	90
4.3	Research Question 2	91
4.4	Efficient and Effective Sampling-based Planner	93
4.4.1	Stable-Sparse-RRT	94
4.4.2	Dynamic Model	99
4.4.3	Experiments	101
4.5	Motion Primitive Formation for Efficient Planning	106
4.5.1	Dynamic Models	107
4.5.2	Stable Maneuver Automaton	111
4.5.3	Learning Motion Primitives	113
4.5.4	Experiments	116
4.6	Summary and Future Work	118
Chapter 5: Risk-aware Planning with Informed Sparse Routing		120

5.1	Introduction	120
5.2	Background	121
5.3	Research Question 3	123
5.4	Risk Reasoning for Safe Flight	124
5.4.1	Risk Under Uncertainty	125
5.4.2	Energy Risk	127
5.4.3	Collision Risk	131
5.4.4	Situational Awareness Risk	132
5.4.5	Combined Risks	135
5.5	Risk-Aware Planning	138
5.5.1	Background	140
5.5.2	Energy-aware SST	140
5.5.3	Informed Energy-Aware SST	142
5.5.4	Informed Multi-Risk-Aware SST for Aerial Disaster Response . . .	151
5.6	Summary and Future Work	154
Chapter 6: Disaster Situational Awareness from Aerial Trajectory Planning . .		156
6.1	Introduction	156
6.1.1	Disaster Response Areas of Interest	157
6.2	Multi-Goal iRASST	158
6.3	Final Demonstration	159
6.4	Summary and Future Work	162
Chapter 7: Conclusion and Future Work		164

7.1	Summary of Research Questions	165
7.2	Summary of Contributions	170
7.2.1	Framework and Tools	170
7.2.2	Benchmark and Demonstrations	171
7.3	Next Steps and Future Work	172
Appendices		174
	Appendix A: Additional Theory	175
	Appendix B: Supplementary Results	182
	Appendix C: Key Software Resources	215
References		216
Vita		232

LIST OF TABLES

1.1	Major Disaster Events since the Start of this Thesis	3
1.2	The Deadliest and Costliest Recent Disaster Events in the United States . . .	3
2.1	Planning Selections and Pros and Cons	26
2.2	Seminal Thesis Works and their Contributions and Gaps	30
3.1	Map Data Options	40
3.2	Radial Basis Function Parameters	45
3.3	Terrain Model Comparison	48
3.4	Comparison of Brute Force and K-Nearest-Neighbors search whether a position is occupied in the map	50
3.5	Structure Discretization Experiment using Washington DC Boundary Zone .	51
3.6	Aerial Dataset Options	63
3.7	Geometric Information of Data Zones for Training and Evaluating Network	66
3.8	Class Levels	73
3.9	Evaluation Metrics for Semantic Segmentation Predictions	74
3.10	Prediction Set Performance Metrics for Level-1 Classes with RGB Labels .	74
3.11	Pix2Pix RGB to 5 Labels Confusion Matrix Values	74
3.12	Evaluation Metrics for RGB Output Pix2Pix Model for Level 2 Classes (5 Labels)	75

3.13	Evaluation Metrics for 5-Channel One-hot Encoding Output Pix2Pix Model for Level 2 Classes (5 Labels)	77
4.1	SST Parameters	96
4.2	k-Nearest Neighbor Experiment for Time Complexity for 100,000 nodes with 90% active, k=20 and r=250	98
4.3	RRT vs. SST in Random Obstacle Maps Performance Comparison	103
4.4	RRT and SST Solution Probability for 2000 iterations for Two Urban Maps Locations	105
4.5	DJI M100 Parameters	110
4.6	Random Control Target Motion Primitive Parameters	111
5.1	Definitions of Risk Terms	125
5.2	Energy Risk Function Parameters Selected	128
5.3	Collision Risk Function Parameters Selected	132
5.4	SST, iEASST, and EASST Success Rate on 3D generated maps	146
5.5	SST, iEASST, and EASST performance on urban maps	149
5.6	SST, iEASST, and EASST Energy and Flight Time in the Atlanta urban map	149
5.7	Path Performance Results for iRASST	154
5.8	Planning Algorithms Playbook	154
5.9	Planner Playbook on Urban Maps	154
B.1	RRT vs. SST for Random 3D Map Trajectory Search, Empty cells indicate searches that failed to find a path	183
B.2	SST, EASST, iEASST Performance Comparison	184

LIST OF FIGURES

1.1	Emergency Response Situational Awareness	6
1.2	Disaster Response Architecture and Current Areas of Improvement	6
1.3	Drone Disaster Response. Graphic produced with the approval of the creator, Alexander Parmley, from http://www.aparmley.com	10
1.4	A trajectory planner seeks to find a series of states, x , that avoid obstacles from time, t_0 to t_f , with potentially infinite paths, p^i	19
1.5	Internal and External Risks for UAS Operations. Graphics reproduced with the approval of the creator, Alexander Parmley, from http://www.aparmley.com	22
2.1	Global Planning in Washington DC 2D occupancy grid with A* and RRT*, from [48]	26
2.2	Final Implementation Concept	31
3.1	Previous demonstrations of Urban Map Models from the author's previous works in [75] and [46]	35
3.2	Digital Elevation model (DEM) and structure height data for 3D height map. Left side: Grid resolution reduced 10x through resampling; Right side: Grid resolution for 1 km cells	37
3.3	RUM Framework	39
3.4	Example for Residual and Error	42
3.5	Computation Time and Residual for B-Spline Training	43
3.6	Computation Time and Residual for NURBS Training	44

3.7	Computation Time and Residual for GP Training	46
3.8	Terrain Model Surface Visualizations on LA	47
3.9	2D Slice of Terrain Models in X-Z	48
3.10	Building Discretization and Height Formation	50
3.11	Structure Discretization Experiment using Washington DC Boundary Zone .	51
3.12	Changing Volume and Computational Time for Differential Structure Ap- proximations	52
3.13	Bounding Box Buffer on Buildings and Terrain Validation	53
3.14	Discretization Impact on Buffer	53
3.15	Rapid Urban Map Model Examples	56
3.16	Rapid Urban Map Model of Atlanta, Georgia	57
3.17	Difference in Aerial Views of Atlanta (Westside and Midtown) circa 2015 to 2019 from NAIP imagery	58
3.18	Comparison of Labels and Resolution of NLCD, GDW, and RUM Land- cover Mapping in Washington D.C.	59
3.19	Previous Applications of the U-NET Architecture from [100] and [75] . . .	61
3.20	Pix2Pix Model	64
3.21	Process and Tools used for Storage, Prediction, and Visualization	65
3.22	Urban Cover Labels for DC Area	67
3.23	DC Channels	68
3.24	DC Classes	68
3.25	NYC Channels	68
3.26	NYC Classes	69
3.27	Miami Channels	69

3.28	Miami Classes	69
3.29	Results for Pix2Pix Model on Prediction Data using RGB Encoding	72
3.30	Mapping RGB Embedded Labels to Classes	73
3.31	Mapping Samples of Test Data to Classes in the RGB Encoding Space . . .	76
3.32	Comparison of labels for NLCD (top left), Impervious Labels for NLCD (top right), Google Dynamic World (bottom left), and Custom RUM Labels (bottom right)	78
3.33	Atlanta Model for 3D Structures and Urban Feature Labels in FME Software	79
3.34	Atlanta 5 Channel Predictions Histogram	79
3.35	Pipeline for Evaluation of Urban Map Model using FME Benchmark	80
4.1	Planning Experiment Overview	93
4.2	Trajectory generation for VTOL vehicles using SST from [131]	95
4.3	RRT and SST Performance in Verification Experiment, RRT nodes and path in yellow and SST nodes and path in blue	102
4.4	RRT vs. SST in Random 3D Environment with Comparisons of Time, Number of Nodes, and Path Cost	103
4.5	RRT vs. SST Multiple Runs in Random Obstacle Maps	103
4.6	RRT vs. SST Hole in the Walls	104
4.7	RRT vs. SST Comparison for Two Urban Map Locations	105
4.8	RRT vs. SST Trajectories through Terrain and Structures for Los Angeles (Left) and New York City (Right)	106
4.9	Quadcopter Model for Flight Simulation	109
4.10	DJI M100 Hover Data for Quadcopter Model compared to Experimental Data	111
4.11	Energy, Flight Path, and Runtime Performance for 5 Motion Primitives with Different Time Step (dt) Values	112

4.12	Motion Primitives Indicating Open Loop (Green) or Closed Loop (Yellow) Stability	113
4.13	Motion Primitive Solution showing the Original Trajectory in Green and DDP Improved Trajectory in Blue	117
4.14	Motion Primitive Lattice	117
4.15	Empty Map and Atlanta Urban Map Trajectory Results using Motion Primitive Control Actions	118
5.1	Combining Data Layers for Riskmap or Costmap in [48]	123
5.2	Conditional Value at Risk for Distributions with percentile α	126
5.3	Energy Risk Function with varying γ	129
5.4	Energy Risk Function	129
5.5	Gaussian Random Variable, \mathbf{X} and Risk Distribution, $G(\mathbf{X})$	130
5.6	Distribution of Energy and Resulting CVaR Risk Value, scaled to a max of 1	130
5.7	Distribution of Risk and Values of CVaR and VaR from input $X \sim \mathcal{N}(40, 5^2)$	131
5.8	Open-Loop Dynamics under Wind and Noise Disturbances	133
5.9	Collision Risk Function	134
5.10	Distance Normal Distribution and resulting Risk Distribution and CVaR	134
5.11	Viewpoint Knowledge Risk Reduction Function	135
5.12	Viewpoint Risk Function	136
5.13	Risks Summary	137
5.14	Costmap for the Risk of Landing when Risk-aware Planning is applied to a Rescue Mission. Graphics reproduced with the approval of the creator, Alexander Parmley, from http://www.aparmley.com	139
5.15	Mean Energy and one and three σ for Flight through Generated Map	141

5.16	Reflying Paths and Tracked Energy for SST and EASST Algorithms	142
5.17	SST, iEASST, and EASST performance on 3D generated maps	147
5.18	SST, IEASST, EASST Energy and Flight Time in urban maps	149
5.19	SST, iEASST, and EASST performance in the Atlanta urban map	150
5.20	iEASST, and iRASST, and SST in New York City demonstrating all risk metrics	153
6.1	Final Demo in Atlanta with Flood Zones in Blue and Structures in Purple, and the mapped landmark points as pink circles	157
6.2	Final Demo in Randomly Generated Map	160
6.3	Visualization of Path for Atlanta, GT Campus Flood Zones	161
6.4	Risk and Energy for Path	161
6.5	Overview of Final Demonstration Experiment	163
7.1	The SAFER Software Environment	171
7.2	Final Demonstration Flight Waypoints in Atlanta, Visualized in Google Earth	172
A.1	Fitting Data to Distribution, Visualization with the Python Package Fitter . .	176
B.1	Generator and Discriminator Models Respectively, Produced by Tensorflow	212
B.2	Training Results for 5-Channel One-hot Encoding	213
B.3	Mapping Between Labels of NLCD and Dynamic World to Custom RUM Labeling	214

LIST OF ACRONYMS

3D	Three-Dimensional
BVLOS	Beyond Visual Line of Sight
cityGML	City Geography Markup Language
CNN	Convolutional Neural Network
CVaR	Conditional-Value-at-Risk
DDP	Differential Dynamic Programming
DEM	Digital Elevation Model
DHS	Department of Homeland Security
EASST	Energy-Aware SST
FEMA	Federal Emergency Management Agency
GAN	Generative Adversarial Network
GP	Gaussian Process
iDRASST	Informed Disaster Risk-Aware SST
iEASST	Informed Energy-Aware SST
iRASST	Informed Risk-Aware SST
KNN	K-Nearest-Neighbors
MP	Motion Primitive
NAIP	National Agriculture Imagery Program
NOAA	National Oceanic and Atmospheric Administration
NURBS	Non-uniform Rational Basis Spline
OSM	OpenStreetMap
RGB	Red, Green, Blue

RRT Rapidly Exploring Random Tree

RTH Return-To-Home

RUM Rapid Urban Mapping

SAR Search And Rescue

SOC State of Charge

SST Stable Sparse RRT

UAS Unmanned Aerial System

UAV Unmanned Aerial Vehicle

VaR Value-at-Risk

VTOL Vertical Take Off and Landing

W-Hr Watt-Hour

SUMMARY

Over the last decade, natural and climate-related disasters have caused the deaths of over 400,000 people internationally and around one trillion dollars of damage in the United States alone. Researchers predict these numbers will rise due to the impacts of climate change and urbanization. This has led to an increased demand for effective and efficient disaster response that will decrease the loss of life and cost of response. One area of improvement for first responder success is in the area of situational awareness. First responders have limited situational awareness of the environment, limiting the actionable information to decide where to go and how to get there. There is a need for efficient data storage and effective algorithms to explore the 3D urban environments during disasters so first responders can coordinate and navigate safely and consistently.

External sensing is one solution that has been used in the past to help first responders make informed decisions. In particular, autonomous unmanned aerial systems have proven the capability to provide support and situational awareness during disasters in the past thirty years. Small aerial systems can now operate under challenging environments and perform exceptionally, partly because of the advancements in battery energy storage, sensor and flight controller weight, and data-driven decision-making. However, recent disasters have shown that they are still limited in flight by guidance constraints. The systems must be capable of rapid monitoring over large areas and in challenging stochastic environments. The two most significant concerns for autonomous aerial systems are limited energy and uncertain obstructions.

Recent work has sought to address these limitations in guidance performance by improving path planning, control, and state estimation. However, unanswered questions remain regarding the ability of these systems to respond efficiently and effectively in low-altitude, 3D flight over large areas. Previous literature has compared different trajectory planning algorithms like RRT* and A* and evaluated the sensitivity to the dimensionality

and scale of the planning problem to minimize the computational time and optimize the path before deployment of the systems. However, the literature lacks a detailed investigation into the performance of trajectory planning algorithms in urban environments with the assumptions of 3D flight, low-altitude navigation, and multiple risk metrics to maximize flight safety and mission success.

Furthermore, many current research topics are focused on the onboard decision-making and planning of these vehicles to account for the unknown. However, disaster events are challenging to model and computationally expensive to simulate. Therefore, this work seeks to develop a framework and software environment to investigate the requirements for offline planning algorithms and risk models to prepare for disaster response and active situational awareness. These strategies and tools can be used by first responders to evaluate how to best prepare for scenarios and to produce flight plans in real-time during disaster events.

The following work seeks to address gaps in literature through the development of a framework for aerial system situational awareness for evaluating and comparing algorithms, datasets, and scenarios. The framework provides tools to evaluate new methods against benchmarks and forms a new benchmark for evaluating mapping and planning algorithms. The primary contributions are systematic improvements to the rapid mapping of urban environments and the high-dimensional trajectory planning in the urban maps. The computationally efficient urban model methodology is developed by optimizing techniques and sourcing the best datasets. A 3D, kinodynamic constrained, risk-aware planner is developed using a sampling-based planning approach and formal reasoning of urban risk metrics.

Experiments investigate four stages in the modeling and simulation environment and determine the best data structures, algorithms, and parameters for optimal performance. A quadcopter is assumed to be the vehicle of interest, and a set of cities are selected for experimentation. The experiments build upon one another leading to a final experiment

demonstrating the models and algorithms' success in preparing aerial system trajectories. The four stages are mapped from the four research questions that seek to address four items for aerial system situational awareness in disaster response: decision-making, path planning, safe flight, and experimentation.

First, a rapid urban map is needed to plan accurate paths through the environment. Therefore, an investigation into data structures and algorithms for modeling the terrain and obstacles is performed. The most efficient and informative modeling methods are selected, which include a Gaussian Process used for terrain models and a geospatial data processing environment for matching buildings and other structures to the terrain. The new rapid urban map methodology is evaluated using specific toolsets against benchmark maps stored at cityJSON files. The framework can produce and visualize any region of interest with accessible terrain and structure data. However, environmental changes may cause available datasets to become inaccurate. Therefore, a deep learning approach is selected for urban landcover prediction using satellite imagery that provides updated and detailed obstacle maps. Furthermore, this additional step provides new labels of the environment that is useful for disaster response.

Second, an algorithm is selected for high-dimensional, kinodynamic trajectory planning in 3D urban maps. Sampling-based methods have successfully solved high-dimensional motion-planning problems through stochastic exploration. Current limitations exist for trajectory planning with good finite-time performance and in large maps that leverage black-box dynamic models. The assumption is made that expensive dynamic models are necessary for the prediction of energy while maneuvering in windy conditions. Therefore, forward-propagation dynamics models or black-box dynamic functions are used, with a quadcopter model selected for experiments. The Stable-Sparse-RRT, SST, algorithm is selected as an improvement to the original kinodynamic RRT algorithm with additions that promote sparsity and convergence while keeping near-optimality guarantees. SST has shown successful performance for robots and aerial systems for finite-time performance.

Next, the computational limitations for finite-time solutions are improved by investigating alternative forward-propagation methods for the black-box dynamic models. The development of motion primitives is demonstrated using a rapid trajectory planner from an acceleration-based thrust and attitude setpoint quadcopter model. The differential dynamic programming algorithm is used to iteratively improve a nominal trajectory with a smooth and stable set of controls.

Third, a risk-aware planning approach is leveraged to provide confidence in mission success. Risks are defined as internal or external uncertainties that can be mapped to the probability of mission success. The risk metrics include the concern of running out of energy in flight updated with a Kalman Filter, the potential for collision with an obstacle or the terrain, and the failure to collect sensor data on essential targets. The three risk metrics are mapped through exponential risk functions. Energy and collision risks are stored as distributions and mapped to a single risk value using the statistical measure of Conditional-Value-at-Risk (CVaR). Alternatively, the viewpoint risk is inversely mapped to a knowledge function with predefined thresholds. The CVaR and viewpoint risk metrics are combined into a single quantitative metric relating to the minimal likelihood of worst-case scenarios. Heuristics are included for each risk metric to provide information that improves convergence and paths in finite-time performance using the informed SST algorithm additions.

Lastly, the three modules are combined into a framework with rapid urban maps and a risk-aware planner. The framework's capability is evaluated against benchmarks for mission success, performance, and speed while creating a unique set of benchmarks from open-source data and software. A series of experiments demonstrate the mission success and performance improvements of the planner, and the simulation environment provides insight into offline planning limitations through Monte Carlo simulation with environment wind and system dynamics noise. Additions are made to the planner for multi-target planning that improves the ability to visit multiple goal locations and then return home before running out of battery energy. The framework is demonstrated in Atlanta, Georgia, in a

flood scenario where likely flood zones are used as landmarks for situational awareness. The final algorithm, Informed Risk-aware SST, iRASST, is compared to other algorithms in the final demonstration and shows improvements across all performance metrics.

The contribution of the work is a framework for advancing planning algorithms and formal risk reasoning for similar problems in complex, 3D environments. The framework is developed through three modules. One, rapid urban mapping methods are evaluated to balance computational cost and accuracy. Two, an efficient planning algorithm explores the rapid urban map in a simulation environment capable of metrics for aerial vehicles in low-altitude, 3D flight. Three, the planning algorithm is reinforced with risk-aware exploration that maximizes the chance of mission success when considering energy, collisions, and situational awareness data acquisition, specifically in disaster response scenarios.

The efficient and kinodynamic constrained trajectory planner with risk-minimizing objectives is proven to provide consistent situational awareness to first responders more effectively and efficiently than previously available methods. The mission success, completion time, and sensor outputs are compared against benchmark algorithms. The framework and software environment are made available to use as benchmarks in the field with the hope that this serves as a foundation for increasing the effectiveness of first responders in the already challenging task of urban disaster response.

INTRODUCTION TO AERIAL SYSTEM DISASTER RESPONSE

Recent data shows the immense human cost of both natural events and violent conflicts. For example, in the past three years of the National Oceanic and Atmospheric Administration's documented major disasters, there have been 50 events causing 553 deaths and costing \$243.3 billion. For the majority of the writing of this thesis, the COVID-19 pandemic has been plaguing the world and is still disrupting people's lives. Furthermore, there were over 100 disasters during the first six months of the COVID-19 pandemic, at least ten of which impacted over 250,000 people [1]. Other significant natural disasters, such as the California wildfires of Summer 2020 and the Texas winter storm of February 2021, occurred during this period. Additionally, domestic and military conflicts in Syria, Afghanistan, Azerbaijan, and Ukraine have threatened lives globally. Additional events of the past three years are summarized in Table 1.1.

1.1 Disaster Events

A disaster is defined as a severe event causing significant destruction that prevents a community from functioning. The word disaster comes from the French word *désastre* and the Italian word *disastro*, which translates roughly to 'ill-starred', or perhaps more appropriately to "not favored by the stars."¹ In modern English, the word disaster is often used regarding some horrible event that impacts one or more people. The consumption of disasters as entertainment has increased dramatically in pop culture, especially in cinema: natural disasters serve as both setting and antagonist in Hollywood blockbusters such as "2012," "Twister," and "The Day After Tomorrow." This interest in the cinematic depiction of these events is paralleled by Americans becoming simultaneously more aware of and

¹<https://www.etymonline.com/word/disaster>

prepared for disasters, according to the Federal Emergency Management Agency’s disaster preparedness survey². For years, government and non-governmental organizations have sought to better understand and prepare for disasters. However, the ability of organizations to prepare and respond to disaster events is diminished by the inherent and chaotic unpredictability of both natural and man-made disasters.

1.1.1 Types of Major Disasters

A disaster event can be caused by the natural shifts within the planet, by a deliberate act of destruction, or by an act of negligence. According to Murphy in [2], these are categorized as "Natural Disasters," "Man-made disasters," and "Mining, mineral, or energy-related disasters," respectively. Natural disaster events include flooding, structure collapse, and explosions. These events often occur sequentially or simultaneously, which multiplies confusion and disruption for those affected [3]. For example, extreme weather events can increase the likelihood of floods, avalanches, and forest fires. Similarly, earthquakes and volcanic eruptions can lead to tsunami waves and flooding. Furthermore, as seen during the past three years, global epidemics can reduce first responder availability and delay response times, resulting in even worse human and structural damage.

In 2017, Hurricane Irma, Hurricane Harvey, and Hurricane Maria repeatedly caused havoc on the Gulf Coast and its surrounding region. These powerful storms brought high-speed winds and nonstop rain, a combination that led to flooding and significant structural damage. In 2021, Hurricane Ida caused record-breaking flooding from the southeast to New York City. The infrastructure throughout the United States proved to be unprepared for these events. Meanwhile, local and federal government agencies have had significant challenges investing in disaster resilience [4].

²<https://community.fema.gov/story/2020-NHS-Data-Digest-Summary-Results>

Table 1.1: Major Disaster Events since the Start of this Thesis

Event	Description	Year
Covid-19 Pandemic	Limitation of human interaction and global supply chains	2020-
Beirut Explosion	An explosion at the Port of Beirut caused deaths and destruction	2020
California Wildfires	Relocation and major destruction of structures	2020
Texas Winter Storms	Loss of power and limited movement of residents	2021
Germany / Belgium Flooding	Flooding of cities and loss of life	2021
China's Henan province Flooding	Major flooding of streets	2021
Hurricane Ida (Louisiana)	Major winds and flooding damaging structures and infrastructure	2021
Tropical Storm Ida (New York City)	Subways and streets flooded with water limiting transportation	2021
Seaside Florida Condo Collapse	Loss of life from structural collapse	2021
Earthquake in Haiti	Major destruction of destabilization of region	2021
Russian Invasion of Ukraine	Military strikes causing urban disaster zones	2022-

Table 1.2: The Deadliest and Costliest Recent Disaster Events in the United States

Event	Date	Cost (billions \$)	Deaths
<i>Hurricane Maria</i>	2017	98.1	2,981
<i>Ohio Valley Midwest Tornadoes</i>	2011	12.5	321
<i>Hurricane Irma</i>	2017	54.5	97
<i>Hurricane Harvey</i>	2017	136.3	89
<i>Hurricane Sandy</i>	2012	77.4	159

1.1.2 The Cost of Disasters

According to the International Federation of Red Cross and Red Crescent Societies, over 410,000 people were killed from extreme weather and climate-related disaster events in the past ten years [1]. The National Oceanic and Atmospheric Administration (NOAA) has documented 285 disasters totaling \$1.875 trillion from 1980-2020 [5]. The frequency and fury of these events are increasing, with 56 events causing over 1,000 deaths and \$315 billion of damage between 2019 and 2021. Some of the deadliest and costliest events in recent history are documented in Table 1.2. The Dixie fire required over 6,000 people working around the clock and cost over \$610 million for three months of fire containment³.

In the past ten years, natural disaster events constituted 83% of all major events causing loss of life and property [1]. The dangers of these natural disaster events are only increasing in the 21st century. According to the Intergovernmental Panel on Climate Change, 2021 has provided even more certain data that human influence has warmed the atmosphere and

³<https://www.nytimes.com/interactive/2021/10/11/us/california-wildfires-dixie.html>

that widespread changes are occurring at an increasing rate [6]. In addition, urban areas continue to grow and become more congested [7, 8], raising fears that the cost of lives and damages will only increase in the future [9]. The UN predicts that 58% of the world population will be living in urban areas by 2050⁴. The concerns for the impact of climate change and urbanization on disaster events and emergency response are only growing. In the initial days of the COVID-19 pandemic, there was a theory that urban regions were growing less crowded from people who wanted to avoid dense areas⁵. However, a mid-2021 analysis by the New York Times found that "...as disruptive as the pandemic has been in nearly every aspect of life, it does not appear to have altered the underlying forces shaping which places are thriving or struggling"⁶. It is clear that the growth of urban cities is not stopping anytime soon, not even from life-changing pandemics.

1.2 Disaster Response and Rescue

Although the history of the word *disastro* has connections to an astrological sense of uncontrollable fate, modern onlookers expect government and social organizations to be prepared to prevent and respond to disasters. Current operations involve local and federal agencies coordinating through programs such as Federal Emergency Management Agency (FEMA)'s Urban Search And Rescue (SAR)⁷ or Hurricane Planning and Response⁸ program. FEMA also coordinates with the Department of Homeland Security (DHS) to research new and novel disaster response technology. Other example organizations include the Maui Search and Rescue and San Diego Search and Rescue groups. The access to resources differs from group to group. However, for both government and non-government organizations, the operation cost is accrued in both the monetary cost of operation and in

⁴<https://www.un.org/development/desa/en/news/population/2018-revision-of-world-urbanization-prospects.html>

⁵<https://www.nytimes.com/interactive/2020/05/16/nyregion/nyc-coronavirus-moving-leaving.html>

⁶<https://www.nytimes.com/interactive/2021/04/19/upshot/how-the-pandemic-did-and-didnt-change-moves.html>

⁷<https://www.fema.gov/emergency-managers/national-preparedness/frameworks/urban-search-rescue>

⁸<https://www.fema.gov/emergency-managers/risk-management/hurricanes>

the sense of safety and comfort for the human response teams [10, 1].

First responders and local volunteers often engage in dangerous and complex situations. When it comes to these complex, dirty, and dangerous missions, data must be distributed rapidly [3]. Data acquisition may be needed during search and rescue, medical transportation, or reconnaissance in rural and urban environments [2]. Regardless of the type of mission, disaster situations often make collecting and communicating data extremely difficult. For example, the 9/11 attacks featured many brave men and women heading directly into what was equivalently the mouth of a dragon — an unprecedented and incalculably dangerous situation. The smoke and fire were intense, and overhead debris and hazards could collapse anytime [11]. Since this event, there has been a heightened focus in the United States on advanced technological solutions that decrease the risk to humans and increase the probability of mission success.

1.2.1 Technological Advancements for Disaster Response

No matter the method of destruction, humankind continues to find solutions to live on this ever-changing planet. Humankind's resilience is shown through the modern disaster preparedness and response infrastructure. Formal disaster response often features three phases of structured tasks: prevention, response, and relief. The response phase occurs in both urban and open areas, could involve uncertain environments, and may involve search, reconnaissance, inspection, and general navigation. Practical and trustworthy operations are necessary to improve the response and recovery phase by maximizing the information and communication during the mission [9]. Son in [9] sought to improve the methods for communicating and digesting information in a disaster response scenario through the perception, interpretation, and prediction tasks. Figure 1.1 outlines the flow of information through the disaster response mission and the three key tasks of perception, interpretation, and projection.

Figure 1.2 provides an architecture for how first responders must make decisions with

the flow of information from the environment and previous training. The mission objective and current state are inherently linked to the communication between the responder and the command station. Improvements can and have been made to the response mission through improving communication, data visualization, and training. However, limitations in data accessibility, data processing, and complex decision-making require technological improvements that may or may not exist in the market today. Therefore, research today seeks new approaches to mission planning and novel technology infusion.

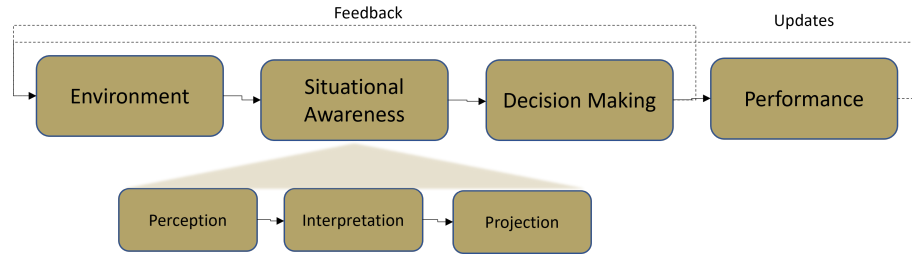


Figure 1.1: Emergency Response Situational Awareness

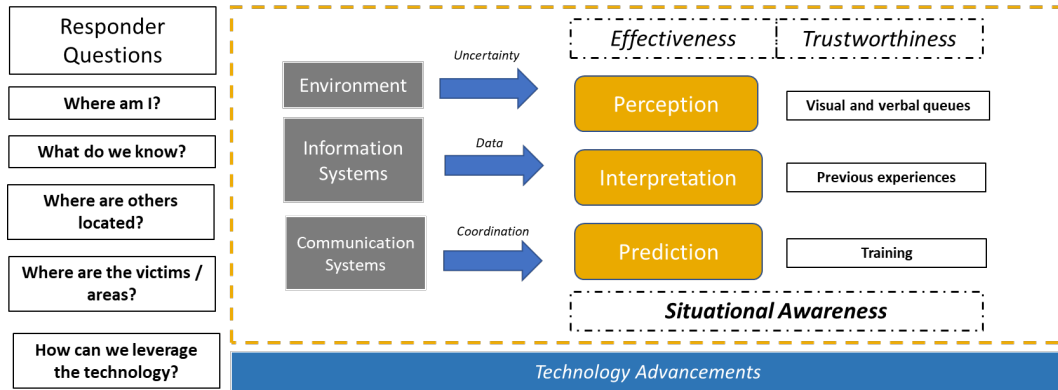


Figure 1.2: Disaster Response Architecture and Current Areas of Improvement

Work in [9] and [12] sought to improve situational awareness of emergency responders through enhanced perception, interpretation, and projection of information. The idea is that the data acquired can be enhanced with better sensing, and the data processing can be upgraded with new tools or algorithms. Improved situational awareness allows first responders to make the best decisions in a disaster scenario when coupled with preparedness and training. Fabbri et al. in [13] investigated risk management and used remote sensing for

disaster response. Cases include mapping infrastructure such as roads for disaster recovery efforts or detecting major hazards for response decisions. In [14], a low-cost UAV is used to assist emergency response by providing high-resolution, real-time imagery with special consideration of the image format and georeferencing parameters.

A 2014 report by the United Nations Office for the Coordination of Humanitarian Efforts (OCHA) detailed the potential of Unmanned Aerial System (UAS) for disaster response by examining case studies in the Philippines, Haiti, and the Democratic Republic of Congo [15]. OCHA examined the capabilities and limitations of UAS for damage surveillance, medical package delivery, and peacekeeping. The responses to these emergency situations are often dangerous tasks. For example, rescuers must use small boats and helicopters to search for lost persons in the region during major flooding events.

One recent example is Hurricane Harvey in Houston, Texas. Remotely piloted UAVs from multiple sources were used for accessing, surveying, and mapping. Similarly, during the Hard Rock building collapse in New Orleans, Louisiana, UAVs were deployed to check the stability of the building before first responders were allowed to enter. Further examples include aerial support conducted after Hurricane Maria, multiple unmanned explorations into the Fukushima nuclear plant following its meltdown, and aerial package delivery of medical supplies during the COVID-19 pandemic. These are a small subset of scenarios in the past decade that have demonstrated the capability of aerial systems for situational awareness in life-saving applications, where the replacement of the human with autonomy and the extension of the eyes through the use of cameras have provided advanced disaster response and search and rescue roles. There is still a critical need for advanced technological solutions that take humans out of danger and complete the mission safely and effectively. These new technologies could improve the life-saving missions of search and rescue, medical delivery, or security surveillance.

Another scenario of interest during the year 2022 is that of wartime response. Russia's invasion of Ukraine has resulted in the mass destruction of urban centers. UAS have

been applied on a decentralized and case-by-case basis. Recent articles about Brinc⁹ have demonstrated some of these examples.

In 2004, Kumar, Rus, and Singh demonstrated in [16] that there is potential for autonomous sensor networks, such as in the case of a burning building. The experiment showed that firefighters could use the external sensing capability but could not actively control them during the mission. In addition, the extremities of the environment clarified that the systems must be able to perceive in smoke and heat. Therefore, more work was still needed in autonomous navigation and active perception. A year later, in [17], Schurr et al. addressed fire and other disaster threats with an integrated team of humans and agents with what they called "adjustable autonomy." Experiments were conducted in simulation using the DEFACTO tool, and the outcomes showed that humans might sometimes diminish the autonomous systems' success. In the 15 years since, a wide range of research has looked into disaster robotics, including Robin Murphy's Disaster Robotics book released in 2014 [2]. This array of experiments concluded that networked autonomous systems could replace humans in many tasks and improve the recovery of humans and supplies during emergency scenarios.

Autonomous systems have thus been leveraged for first responders and emergency response in the consumer marketplace. In the last year, companies developing these technologies have caught the eyes of investors, such as when BRINC¹⁰ raised a 25 million dollar series A funding. Clearly, the technical and commercial success has made UAS a viable option for providing situational awareness during disaster response.

1.3 Aerial and Autonomous Systems

UAS continue to rapidly expand in use across many application areas such as package delivery, construction surveillance, and disaster response [18]. UAS are suited well for these "dull, dirty, dangerous" missions because of their small footprint and agile flight capabil-

⁹<https://www.washingtonpost.com/world/2022/03/24/ukraine-war-drones/>

¹⁰<https://brincdrones.com/>

ities. Their advanced use has benefited from advancements in onboard sensing, control algorithms, and communication architectures that allow for complex flight and estimation in real-time. UAS are often employed in these scenarios to remove the human from the "loop," thus improving the safety and consistency of various tasks within the missions described earlier.

The success of UAS is built on the advancements in MEMs, GPS localization, compute power, and electric propulsion. This has led to the explosion of their design, development, and demonstration showing off the skills of efficient, agile, and autonomous flight.

Disaster response and search and rescue missions already include a range of aerial systems in operation. This has only increased in recent years with the advancements of small unmanned aerial systems or SUAS. However, the general use has been by human-controlled helicopters or by various SUAS that have limited capabilities. The application of swarm UAS has been applied in less critical scenarios but has proven successful in distributing sensing. Therefore, the technological advancements of these systems may be the key to improving the mission effectiveness of rescue teams and local first responders.

Autonomous aerial and ground systems have been successfully applied in many scenarios, including SAR and disaster response [2, 19, 9, 20, 21, 22, 23, 24, 25, 15]. Also, recent work has utilized swarms of UAVs for railway evacuation in [26]. A higher-level investigation of the integration of UAS for emergency management is examined in [27].

The use of autonomous robotics and unmanned aerial systems for search and exploration has been an area of interest for many years. In some cases, these systems have provided support, such as the four-legged robots that searched the Fukushima Daiichi nuclear power plant¹¹. There are still many problems to be solved in this area since UAS must be able to operate in complex conditions, see Figure 1.3. These scenarios require advanced capabilities such as operating in GPS-denied, limited-vision, or high-sensor noise scenarios. Weather, obstacles and adversarial agents are some difficulties for UAS operat-

¹¹<https://www.nytimes.com/2017/11/19/science/japan-fukushima-nuclear-meltdown-fuel.html>

ing in these applications. The operation of autonomous systems at a small scale and cost requires tradeoffs between flexibility and capability, cost and efficiency, safety and speed, and performance and consistency.

Additional limitations and complications exist for emergency response, however. The systems must be able to make autonomous decisions in difficult situations with the potential for extended periods. In addition, the size, speed, and cost of these systems must be reduced because of both the platform limitations and the cost-limited critical response timeline. "Unmanned aircraft can fly lower than traditional aircraft and achieve the same if not better quality data at a lower cost for small to medium sized surveys," stated the US Department of Transportation¹². Furthermore, smaller aerial systems have demonstrated their ability to collect improved data for planning, mapping, and surveillance. Therefore, the assumption is made that the smaller aerial systems will lower the overall cost, but further independent research is needed to investigate the cost of deploying multiple systems and how this cost compares to traditional methods with larger manned aircraft.



Figure 1.3: Drone Disaster Response. Graphic produced with the approval of the creator, Alexander Parmley, from <http://www.aparmley.com>

The history of aerial systems hinges on the advancements of the reduced platform, the propulsion and electrical energy components, and the improved computational power of CPUs and GPUs. The primary interest in aerial systems for emergency response and disaster relief has come from government entities such as FEMA and DHS in the United

¹²<https://www.fhwa.dot.gov/uas/resources/hif20034.pdf>

States. For example, FEMA has partnered with the Florida State University Center for Disaster Risk Policy since 2017, with the most recent funding applied to the Surfside condo collapse.

The application of these systems has varied from emergency response, search and rescue, and medical transportation [28] and the demonstration of this concept has already been tested in the marketplace through examples such as Auterion for medical supplies delivery¹³, EHang for fire rescue¹⁴, Skydio for situational awareness¹⁵ and Kitty Hawk for emergency response¹⁶. The interest in these systems in the marketplace has been built from the years of research into autonomy and robotics for disaster response. Non-governmental organizations and non-profits such as the Small UAVG Coalition and UAViators are also investing in aerial systems for humanitarian applications since there may be a delay in the technologies being sold for this less lucrative application. Much of the progress for modern use has come from academia, specifically through the robotics and aerospace communities. Research into these topics in the modern sense stretches back to 1992 in AI and Cognitive Science [29], where multiple UAVs were used to collect data for FEMA. Key researchers in this field have been making advancements since 2004 [16]. Recent events such as the International Conference on Robotics and Automation for Humanitarian Applications, the World Robot Summit 2020, and the 2021 IEEE International Symposium on Safety, Security, and Rescue Robotics have pushed the boundaries of the research to real-world scenarios with new algorithms and datasets being published. The foundations can be found in the Disaster Robotics Book [2, 23] and Geological Disaster Monitoring Based on Sensor Networks in 2019 [30].

There is a wide range of literature on aerial systems' application to search or naviga-

¹³<https://auterion.com/auterion-partner-quantum-systems-performs-first-ever-medical-vtol-flight-downtown-in-a-large-city-delivering-covid-19-samples-to-a-laboratory/>

¹⁴<https://www.ainonline.com/aviation-news/business-aviation/2020-08-03/emergency-first-response-could-boost-urban-air-mobility>

¹⁵<https://www.skydio.com/blog/skydio-drone-autonomy-for-situational-awareness/>

¹⁶<https://evtolinsights.com/2021/02/kitty-hawk-and-falck-to-explore-possibility-of-using-heavy-side-evtol-aircraft-for-emergency-response-missions>

tion cases. Autonomous aerial and ground systems have successfully performed in various scenarios, including search and rescue and disaster response [2, 19, 9, 20, 21, 22, 23, 24, 25, 15]. UAS has been tested and studied for search and rescue applications, package delivery, GPS-denied navigation, crowded navigation, and autonomous landing, in addition to many other applications. The implementations have included lidar and camera sensors, SLAM and clustering algorithms, and many different planning and control schemes. Not only have extensive tests been done in simulation, but also on hardware in real-world environments. For example, Lingqvist et al. in [31] demonstrated a subterranean search and rescue mission leveraging point cloud data and a camera stream for localization and reactive local planning. Greenwood et al. in [25] deployed a UAV after the hurricanes in 2017 to evaluate the ability to assess the damage.

In recent years, aerial systems have been used for search and rescue and during the collapse of structures. For example, during the Surfside condo response, aerial systems were used for producing two-dimensional and three-dimensional maps. Mapping and photogrammetry have been the most common use for aerial systems in the past ten years, and extensive research has gone on to improve the accuracy and fidelity of the solutions [32]. Classical photogrammetry methods have utilized ground control points to calibrate and align the aerial images; however, the advancements in inertial sensors and computer vision methods have led to current results using the localization of the camera sensor to calibrate. Then, feature matching methods are used that are robust to noise in the alignment measurements. Success in this application has led to mapping software and sensors being widely available to the general public.

The primary use for UAS during search and rescue is as an extension of human vision. Extending sight provides improved situational awareness, as discussed in the previous chapter. Aerial systems have only recently been allowed to fly Beyond Visual Line of Sight (BVLOS), but the ability to stream video and send commands from afar has been achieved

for many applications. In fact, the FAA has waivers to approve this operation¹⁷, and NASA is continuing to invest in the future of flight corridors and UAS traffic management¹⁸.

For continued advancements, there is a consensus that more autonomous or semi-autonomous behaviors will be required to provide safer, more efficient responses. Examples of this are detailed in [20, 22, 33, 24]. The autonomy of robotic systems has advanced rapidly in recent years with improvements to onboard sensing and artificial intelligence. However, as demonstrated in self-driving vehicles and urban air mobility, the in-field requirements that are set must be high. Progress continues, and Atkins has defined four levels of automation for Advanced Air Mobility (AAM)¹⁹, which are "No capability, Limited, Nominal, Comprehensive." Autonomy represents a system's ability to reason and make decisions, requiring some level of intelligence formed from sorting through data and finding rules or patterns [34].

According to Murphy [2], the capability of these autonomous systems must include the ability to operate in unpredictable environments, navigate in wireless-denied environments, and coordinate in a 'tactical, organic system.' In work by Scherer [35], the task of low-altitude navigation and obstacle avoidance is given major attention as a capability of aerial systems. Choi in [36] demonstrated a two-stage process for obstacle avoidance in crowded urban environments and detailed the computational limitations and complexity of accurate models in simulations. Donato in [37] was focused on emergency landing for rotorcraft; however, it provides insight into the data fusion for predicting the risk in the environment. Pfieffer in [38] sought to use ground vehicles for navigating through an unknown environment, demonstrating the amount of data and testing required to prove successful, even in situations without the requirements of emergency scenarios. Overall, UAS must be capable of quick responses, effective decisions, safe maneuvers, and autonomous operations.

¹⁷https://www.faa.gov/uas/public_safety_gov/public_safety_toolkit/media/TBVLOS_Waiver_Final.pdf

¹⁸<https://www.nasa.gov/press-release/langley/nasa-enters-space-act-agreement-with-longbow>

¹⁹https://nari.arc.nasa.gov/sites/default/files/attachments/atkins_AAM_panel.pdf

1.4 Research Gaps

Disaster response can be improved by further leveraging the autonomy and athleticism of UAS for first responder situational awareness. Current systems exist to be piloted by first responders, and academic work has been interested in aerobatic and efficient flight planning. However, a systematic investigation of the best data processing and planning algorithms has not been studied. Therefore, the overarching research objective of this work is to determine the optimal application of UAS to autonomously monitor disaster environments for situational awareness through autonomous navigation and decision-making.

There remain critical challenges that current systems fail to address completely. UAVs are limited in their ability to fly over large areas because of the energy cost of the flight. Crowded environments have been addressed by fast onboard decision-making and avoidance; however, the system is often required to balance multiple tasks and objectives while also performing reactive avoidance. Low-altitude flight, in general, involves more uncertainty than when flying at a high altitude. In addition, any prior information of the map has now become inaccurate and less reliable.

The use of aerial systems for these emergency response scenarios requires detailed planning at multiple levels of implementation and a range of different algorithms. For example, system-level dynamics, sensors, and power requirements must all be considered. In addition, the algorithms for control and path planning must consider the system-level and operations-level requirements. All this must occur under a high level of autonomy to keep humans out of danger and to maximize mission success. Many of these scenarios require operating with high levels of uncertainty and in complex regions for navigation. Therefore, UAS must navigate and make decisions in these complex regions while considering speed and efficiency at the system level to assist in these critical, life-saving missions.

1.4.1 Current State of the Art

In the current commercial or defense market, there is a limited number of aerial systems for accomplishing missions exactly or similar to the disaster scenarios detailed in previous sections. Research is active for navigating complex, unknown environments like indoor, GPS-denied environments or crowded urban streets. The current state of the art for small unmanned systems is the limited success of companies like Parrot, DJI, Skydio, and Shield AI. Parrot and DJI provide constrained cases of obstacle avoidance for the average consumer taking pictures or doing photogrammetry. Skydio is using multiple cameras onboard to track objects and avoid others from all directions, applying first to cinematography but now showing use for first responders. Shield AI provides similar results for the United States military when service members explore an enemy structure. In addition, Edgybees has demonstrated the success of using aerial systems and cameras for situational awareness. Near-Earth Autonomy and Daedeleon have shown how cameras and other sensors can be combined to find safe landing zones and improve state estimation.

A range of questions remains regarding how to advance these systems to the required level of operations needed. Most importantly, can these advanced sensing techniques and autonomous agents provide better situational awareness and successfully move first responders out of danger? For navigating the environment, what planning algorithms are necessary for efficient flight, and how can this be investigated in a modeling and simulation environment?

This research focuses on four key subjects of autonomous UAS situational awareness. One, flight planning is limited by the computation and data available offline after a disaster event. Large amounts of data are required, and flight planning must be available over large areas of space. For example, consider that the scale of the California Wildfires in 2021 was over seven thousand square miles and that most urban centers in the United States have high-density population centers over 50 to 300 square miles. Two, decision-making requires informative actions from the available data of the environment. Third, safe flight

ensures that mission-affecting failure events do not occur and that there is a high level of confidence. Lastly, experimentation through concise and productive modeling and Monte Carlo simulation provides insight into the success of future real-world scenarios.

1.4.2 Trajectory Planning

LaValle, in the book **Planning Algorithms** [39], details the essential ingredients of the planning problem. Fundamentally, the planning problem is a search problem exploring the set of possible configurations in discrete or continuous state space. Examples of a motion planning problem include the Piano Mover’s Problem or the challenge of the Rubik’s cube. The difficulty lies in the curse of dimensionality of the search space and the computational requirements of solving a path with constraints.

For a problem to be solved, there must be a defined *state* representing the discrete or continuous representation of the *agent*, the vehicle or system of interest. The planning problem is inherently a search problem exploring the set of possible configurations in the state space. The *time* is represented as a discrete or continuous variable to track the changing state of the system. The plan of the agent can be described as *actions*, which are simply the functions transitioning the agent from one state to another. This could be controls of a dynamic system or movements of a puzzle. Of course, the decision of the actions and set of states require the definition of the *initial and goal states* and the *bijective criterion*. The final result, a combined set of states and actions, is the *plan*. Often the critical problem is not necessarily whether a path exists but whether the path is the best as compared through a like-to-like metric with all other paths.

Given a system defined by a state x and an environment decomposed into cells in the same dimensions X . Whether this is considered a set of voxels, also known as grid cells, or a graph, with nodes and edges; the problem can be defined as the search for a path P , which is a sequence of vertices $P = (v_1, v_2, \dots, v_n)$. In this example, assume the vertices are in the same configuration space, X , as the system’s state, so that $v_i \in X \forall i \in (1, n)$. The

problem may have multiple solutions that can be compared through an objective function.

There is also a hierarchy to the path planning problem. This can be divided into global and local planning [35, 40]. Global is generally the high-level planning done before flight from a start to goal location and relies on map knowledge. Local planning occurs inflight and is focused on navigating the environment from waypoint to waypoint, whether for tracking, search and rescue, or following. For example, in [41], Kang explains that "this local trajectory optimization is combined with a global path search algorithm which provides a useful initial guess to the nonlinear optimization solver." Likewise, Choi in [36, 42] a preflight and in-flight distinction is made to plan aerial vehicle flights through an urban environment.

The motion planning problem is often considered when working with agent transitions that are in continuous state spaces. Research into this area has existed since the 1970s, and much interest falls into the field of control theory or dynamic stability. On the other hand, working in discrete spaces can be considered a path planning or simply a search problem. A common framework is to consider discrete states within a continuous world using tree structures like Hidden Markov Models.

Algorithms to solve for feasible or optimal paths differ in mathematical theory and technical implementation. For instance, there is a long history of optimal control problems with both linear and nonlinear dynamic systems. A receding horizon approach is common for onboard planning, as detailed in the tutorial by Mattingley et al. [43]. Alternatively, the problem can be set up through assumptions or linearization as a convex optimization problem. A detailed explanation and tutorial for convex trajectory generation are found in [44]. Advanced methods that enforce convexity can balance the accuracy and computational efficiency of the problem, as detailed in work by Blackmore et al. [45].

Algorithms that operate in the discrete space make additional assumptions on the search space by constraining the resolution of each dimension and limiting the actions that can be made moving from state to state. The first step approach would be to formulate the prob-

lem as a graph, where the system state is a node in a discretized state space. The Breadth First Search (BFS) algorithm can be implemented with a few lines of code to find the optimal problem through recursively solving for all paths. However, the search time complexity is $\mathcal{O}(|V| + |E|)$, where $|V|$ is the number of nodes and $|E|$ is the number of edges. Improvements in convergence start with leveraging dynamic programming and eventually lead to the A* algorithm that improves upon the efficiency of Dijkstra's algorithm by using a heuristic in the cost-to-go function. The total cost function at step n , $f(n)$, is the combination of the running cost, $g(n)$, and the heuristic, $h(n)$, as in Equation 1.1. The algorithm seeks to minimize this by sorting through nodes in a priority queue.

$$f(n) = g(n) + h(n) \quad (1.1)$$

The requirement is that the heuristic function, $h(n)$, is admissible and consistent. An admissible heuristic always overestimates the true distance, $h(n) \geq d(n)$. A consistent heuristic means the optimal path is guaranteed to be found without processing any node more than once, $h(n) \leq c(n, q, n') + h(n')$ where q is the action taken to go to the next node n' . If these parameters are met for the heuristic, then the A* algorithm is guaranteed to be complete, optimal, and efficient, with the fewest nodes needed to find the optimal path. For finite-time performance, it should be noted that no matter the heuristic, the algorithm is optimal. However, the choice affects the number of search nodes required and, thus, the time and memory to compute the solution. For A*, all nodes are stored in the memory, and it requires exponential time to solve.

Kim et al. in [46] addressed UAM trajectory planning using Mixed Integer Linear Programming, MILP, which reduces the problem to a sequential optimization problem. Simplifying assumptions are required, including a single flight altitude and linear dynamics. If the system of interest and the environment for application are unknown or complex alternative methods may be needed. In particular, sampling-based planning through a randomized search algorithm can be used for a discrete search tactic in a continuous state

space. Sampling-based methods leverage a stochastic technique and can guarantee completion and optimality when using particular algorithms. Examples include probabilistic roadmaps or Rapidly exploring Random Tree, RRT [47]. RRT leverages efficient tree data structures, cost-based tree pruning, and random sampling for asymptotically optimal solutions and guaranteed completeness. More details are provided in Chapter 4.

The primary questions that exist when interested in finite-time, real-world performance relate to the dimensionality and resolution of the graph. For example, the potential paths of an aerial vehicle through a 3D environment are shown in Figure 1.4. Initial state $x(t_0)$ and final state $x(t_f)$ are defined in 3D continuous space, with all positions between creating infinite many solutions. Consider two alternative paths, p^1 and p^2 , which avoid the obstacles in two different ways. A sampling-based planner can generate trajectories of this form with a tradeoff in the performance, for instance, whether the path that goes in between the two obstacles can be found by using random search strategies. More details are discussed in Chapter 4.

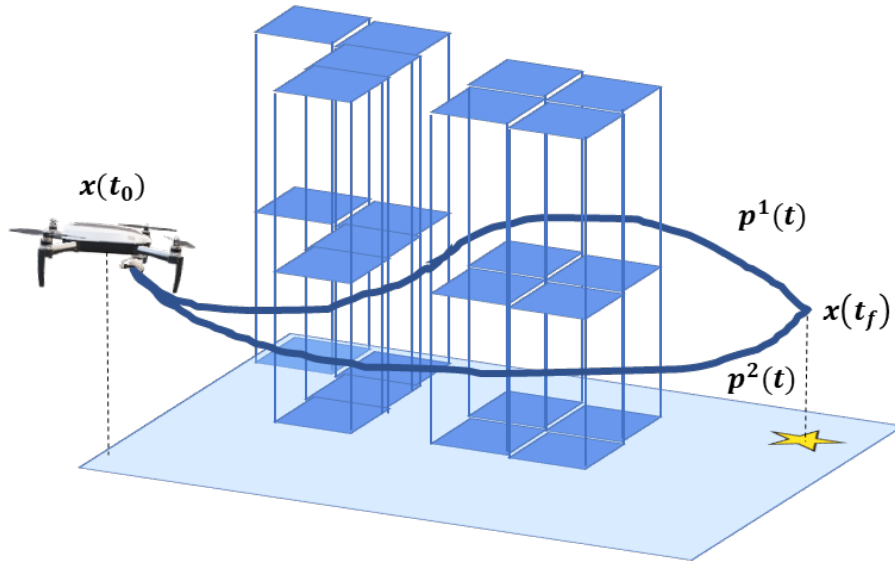


Figure 1.4: A trajectory planner seeks to find a series of states, x , that avoid obstacles from time, t_0 to t_f , with potentially infinite paths, p^i

Motion Planning Background

Consider the generic motion planning problem, where the state space is in X , action/control space in U , and time, t . The objective is to minimize the objective function, J , while satisfying the constraints, g , as seen in Equation 1.2. The objective has a discrete and continuous form, which results in solutions through discrete steps of a differential equations solver.

$$\begin{aligned} J &= \sum_{i=0}^n c(x_i, u_i, t_i) \quad s.t. \quad g(x_i, u_i) < 0 \quad \forall i \\ J &= \int_0^n L(x_i, u_i, t_i) \quad s.t. \quad g(x_i, u_i) < 0 \quad \forall i \end{aligned} \tag{1.2}$$

Trajectory planning for robotics has been regarded as a computationally expensive task because of the many complications that must be considered [23]. For one, robotic systems, whether aerial, ground or otherwise, each have unique uncertainties and constraints from the sensor inputs and kinematics. Also, the difficulty in path planning is with the processing time required to solve and the limited information about the system dynamics and environment. If the dynamics are modeled precisely, and the space is known exactly, then the optimal path can be solved, and the decision is how much time is allowed. This will continue as a tradeoff when more complex scenarios occur, and multiple techniques may be needed for planning an optimal path.

Motion planning has been solved in the past by expensive algorithms requiring exponential time and improvements that led to combinatorial methods of constructing roadmaps. A more straightforward solution has been to define potential fields through a vector field to 'push' or 'pull' the agent through the environment. Recent advancements have increased the use of sampling-based methods. The curse of dimensionality is often recognized as a significant issue if an exact search is required. Therefore, there is usually an interest in keeping the dimensionality small, ignoring dynamic environments, and using heuristic ap-

proaches to approximate the best path. These methods usually have asymptotically optimal paths, for example, the Rapidly Exploring Random Tree (RRT) algorithm [39].

Trajectory planning is critical to performing collision-free, dynamically-feasible flights at low altitudes. If efficient and safe 3-dimensional paths can be planned offline, then overall mission performance will be significantly improved, and the concept of operations for the entire team can be managed more effectively. The choice of the best algorithm and models to use depend particularly on the assumptions and outputs desired and the required computational resources for the algorithms required. Future chapters will explore this in detail.

1.4.3 Decision-Making

There remains a gap in literature on how to prepare for disaster scenario response, other than traditional search strategies. Disaster scenarios are unique and difficult to model. Furthermore, the mission planning for an aerial vehicle has traditionally been controlled in a decentralized manner, meaning that individual pilots make decisions based on local information and communication. However, prior information and simulation studies could provide insight into where the aerials systems should prioritize searching based on the objectives and system constraints.

1.4.4 Safe-Flight

There is now a wide range of applications for aerial systems, from package delivery to urban air mobility, that requires a framework for safety-critical or risk-aware flight planning and motion planning. However, the focus has been on local onboard planning instead of offboard large-scale global planning. The scale and uncertainty of offline planning raise concerns with the previously developed methods. UAS must account for many environment feedback signals and epistemic and aleatoric uncertainties in flight. For example, see Figure 1.5 where risks such as wind, limited power, or obstructions are constantly plaguing

aerial systems.



Figure 1.5: Internal and External Risks for UAS Operations. Graphics reproduced with the approval of the creator, Alexander Parmley, from <http://www.aparmley.com>

1.4.5 Modeling and Simulation

Another important distinction is whether an environment is known or unknown. If the environment is completely known, a map with all information needed to plan an optimal path exists. Therefore, the only requirement is to find a planning algorithm that trades accuracy and computation. However, the more common case, particularly in disaster response scenarios, is an unknown environment. A good example would be someone using a hand-held map of the city. However, new construction or flooding has changed the location of buildings or limited the accessibility of streets. An accurate but efficient modeling and simulation environment is necessary for exploring the tradeoffs and evaluating the performance of the algorithms and data needed to make smart decisions and fly safe trajectories.

The dynamics of the system or the environment can be modeled using chaotic systems or nonlinear dynamic systems. Data and validation techniques improve the confidence that mathematical models represent the systems accurately. A modeling and simulation environment that can access models, algorithms, and data efficiently allows for repeated cases, efficient verification, and complex experiments.

1.5 A Framework for SAFER Response

From Scherer's [35] investigation into low-altitude rotorcraft flight and Pfeiffer's [38] research for map-less navigation in uncertain environments, it can be said that the avoidance of obstacles during navigation is one of, if not the most critical requirement for mission success. This is because of the high level of difficulty in avoiding safely and the catastrophic circumstances if there is a failure to avoid. The question remains how to most effectively and efficiently bring these systems to use for many applications. This requires an understanding of the cost, performance, and logistical limitations. The technological advancements for emergency response in situational awareness and the integration with networked autonomous systems provide the template for implementing a new safe and effective solution. The overarching research objective is therefore defined as the following.

Research Objective

Perform a systematic investigation of the best data and planning algorithms through current technology and tools for unmanned aerial systems to supplement first responders' situational awareness.

This work seeks to develop a modular framework that can be evaluated and tested in multiple scenarios with trade studies and sensitivity analyses. The framework is named SAFER (Safer Autonomy For Emergency Response). The next chapter provides an overview of the SAFER framework and background literature to formulate the current gaps in this research area. Disaster scenarios are difficult to model, and therefore the SAFER framework addresses the flight planning problem entirely offline.

A FRAMEWORK FOR DISASTER RESPONSE WITH UNMANNED AERIAL SYSTEMS

Autonomous or semi-autonomous unmanned aerial systems for emergency response have been explored for over 20 years with work such as in [16, 12, 17]. However, there is a lack of investigation into the current technology that meets safety-critical, real-world scenario requirements. A detailed explanation of aerial systems autonomy is found in [34]. This work investigates a modular framework that can be evaluated and tested in multiple scenarios with sensitivity analysis and Monte Carlo simulation.

The overarching hypothesis is that the effectiveness of aerial systems for providing situational awareness in disaster response is dependent on the trajectory planning, decision making, safe flight prediction tasks, and modeling assumptions. Therefore, the algorithms and data to form a 3D representation of the environment and to plan dynamically-feasible paths must be explored in a simulation environment that must balance the computational load and the accuracy. By modeling the uncertainty, the effectiveness of the models can be explored through Monte Carlo experiments and sensitivity analysis.

2.1 Urban Map Modeling

The environment must be represented to perform a successful plan from the start to the goal. The best setup is an a priori one, where a map with complete knowledge of the state space and obstacles can be referred to as free-space and obstacle-space. The modeling of the environment is detailed extensively in [34]. If a detailed map is unavailable, as is often true in real-world applications, it must be predicted or approximated using data. Examples of this are using point cloud data to build 3-dimensional maps of structures. However, if detailed point cloud data is unavailable, a lower-resolution or lower-fidelity environment

representation may be required. Predicting or approximating maps using a single data source has limitations. Combining multiple sources through decision trees or data fusion can improve accuracy, reliability, and consistency.

2.2 Offline Aerial Flight Planning

Flight planning conducted offline for aerial system trajectories are an area of research with many applications. Examples include trajectory generation, landing exploration, or operations analysis. There are many ways to solve this problem, as introduced in Chapter 1. An overview is seen in Table 2.1.

There are four primary methods to solve the obstacle avoidance problem, and each has its advantages and disadvantages. Stochastic methods assume that a good solution is formed using a randomized path that is iteratively improved. Methods such as Rapidly-exploring Random Tree, RRT, are powerful because of the fast computational time that is not limited by non-convex or high-dimensional spaces. However, RRT may have difficulty finding solutions quickly, and some results may be sub-optimal. Advancements such as RRT* can guarantee optimality, but there are still trade-offs to consider. Potential field methods represent the environment with particles and force fields that cause the system in question to be pushed or pulled around the space. Methods such as Force Maps are simple to set up but do not guarantee collision-free paths and may need to be tuned for various applications. Road map methods utilize a graph or grid structure in the environment to plan paths from a start to a goal node or cell. Algorithms such as the A* algorithm use search heuristics and can guarantee collision-free paths while solving the problem very quickly. The computation time largely depends on the search space size and the heuristic. Geometric or optimization methods solve for guaranteed collision-free paths using the true or approximated geometric information and objective functions. These methods may be complex to implement and are dependent on the types of vehicles and obstacles of the scenario.

Table 2.1: Planning Selections and Pros and Cons

Method	Details	Pros	Cons	Use?
Sampling-based	Stochastic search in continuous space	Flexible with guarantees	Parameter sensitivity	✓
Heuristic Search	Grid search in discrete space	Efficient in low dimensions	Exhaustive without a good heuristic	X
Optimal Control	Continuous dynamics prepared for TPBVP	Accurate and optimal	Computationally expensive and case dependent	X
Optimization	System and constraint relaxations	Fast solvers	Limited accuracy	X

Previous research has investigated planning algorithms for urban map models. For example, a demonstration of two common path planning algorithms investigated in [48] is seen in Figure 2.1. The A* and RRT* algorithms formed different paths, but performed similarly in terms of speed and total distance when applied in a 2D occupancy grid. Ochoa in [49] also investigated individual algorithms in urban environments to begin to understand the capabilities and limitations of trajectory planning algorithms. However, a deeper investigation into the critical requirements through a modeling and simulation environment has been lacking.

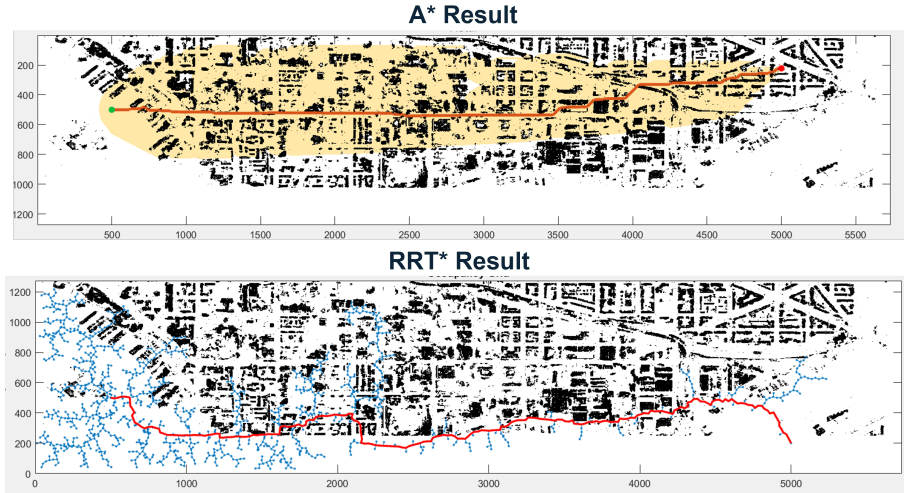


Figure 2.1: Global Planning in Washington DC 2D occupancy grid with A* and RRT*, from [48]

Sampling-based planners are selected as the method for offline aerial flight planning because they are capable of the core capabilities needed. Sampling-based planners are ca-

pable of collision-free trajectories, feasible dynamic solutions, and optimal trajectories with custom objective functions. Furthermore, the randomized search pattern, essentially an indirect shooting search algorithm, allows exploring uncertainty environments and reusing search patterns for updated trajectories. Chapter 4 investigates this in detail while demonstrating the success when using black-box dynamic models and targeting finite-time solution quality.

For complex aerial vehicles, the dynamics are critical to stable flight performance. When considering trajectory planning, the dynamics must play an essential role in finding feasible and optimal plans. For example, in [50], the dynamic constraints must be considered in the planning, and similarly in [51]. In [52], the dynamics are not required to be defined in an equality constraint; instead, a black-box simulation is used to forward propagate that state. All of these techniques require varying forms of computationally expensive processes. In other examples, dimensionality reduction or a linearized form of the dynamics is used to simplify the problem. Additionally, trim states may be defined and explored to find a path within the feasible manifold of the complete configuration space. The maneuver or state is often referred to as a Motion Primitive (MP).

2.3 Safe Flight Under Uncertainty

Feasible or single-metric optimal trajectories from offline aerial flight planning can be challenging to acquire. However, the foundational algorithms have proved successful. Uncertainties in how to plan these paths remain, especially for complex missions in uncertain environments. The question remains on how best to define the objective or how to leverage special measures called *risk metrics*.

Recent research for global path planning and offline planning has been applied to urban air mobility emergency landing and unmanned aerial vehicle flight planning. Hu et al. in [53] linearly combined three risk maps and then compared the Dijkstra, A*, and Ant Colony planning. Primates in [54, 55, 56] and Ding in [57] investigated how risk indicators such

as population density and obstacle data can be used to make decisions on safe flight zones with UAVs or safe landing spots for rotorcraft in the case of an emergency. Geographical and population data was used for offline risk map creation. In a similar study, Hu et al. in [58] ran a sensitivity analysis on parametrized features in a linear combination. Di Donato [59] and Harmsel [60] combined obstacle, terrain, and density maps to form a costmap, while Slama et al. [61] combined obstacle maps with area terrain types to find the least risky paths for emergency landing. Primatesta [55] and others have implemented population risk and categorized selections as the probability of an accident, probability of fatality, sheltering effect, and population density, with the use of population, land use, obstacles, and restricted airspace datasets.

2.4 Recent Contributions to Research

A series of theses from the last 20 years have bridged the gap for autonomous and aerial systems applications in the world today. For example, the consideration of risk has been shown in [62] and [63], but this did not consider sampling-based planning improvements from [35] and [64]. Furthermore, the literature lacks an investigation into this application of autonomous aerial vehicles for urban disaster response along low-altitude and long-distance flights.

Previous work has compared different algorithms one-to-one like RRT, A*, BIT* in [49]; however, this was 2D planning which removes many of the critical constraints from low altitude 3d planning. [65] explored different metrics and tracked them across flight, adding limits and constraints, and [37] used a risk map created from offline population and environment datasets. However, this was not the approach in a formal risk reasoning approach as in [62]. Xiao and others, such as in [66] approached risk metrics in flight planning but have not applied this specifically to disaster response scenarios. Those that have applied to real-world scenarios often demonstrate package delivery or emergency landing where there are different constraints and objectives.

Gaps remain in the field, however. For instance, the concept of risk reasoning must continue to be formalized for urban risk metrics, in particular for risk from both internal states and external hazards. Safety guarantees from offline planning are still limited by the accuracy of modeling and probabilistic methods. Sampling-based planners have shown success with simple distance or energy-based objectives but have not been leveraged for risk metrics in much detail.

2.5 SAFER Framework Contributions

Previous work has explored aerial autonomous systems’ ability to navigate complex environments, but not in a self-contained system of systems framework. Ochoa directly compared planning algorithms RRT, A*, and BIT* in [49]; however, this was in 2-dimensions, which removes many of the critical constraints from low altitude 3d planning. Low altitude 3D planning is focused in more detail by Scherer in [35]. However, the focus is on tracking waypoints from a high-level planner and on landing safely. Ochoa in [49] explored the key metrics to track during flight, but the work was missing a formal risk reasoning approach as was done by Xiao in [62]. Xiao and others have approached risk through the view of probabilistic measures that can guarantee mission success and minimize the change of mission failure under large uncertainties. Other examples include the use of Conditional-Value-at-Risk in [66, 69, 70]. This approach has not yet been applied to the types of environments and uncertainties that occur in disaster response scenarios.

This work develops a modeling and simulation environment using openly available tools and datasets to investigate the safe and efficient aerial response for disaster situational awareness using low-altitude 3D motion planning, rapid urban map models, and formal risk reasoning for safe offline flight plans. The framework for managing the data and flight planning is inspired by previous works such as in [71, 72, 65]. Continuations of the formal risk reasoning are extended from work in [62] and [63], but while applying sampling-based planning improvements for low-altitude urban flight from [35], and [64]. Overall, this

Table 2.2: Seminal Thesis Works and their Contributions and Gaps

Source	Key Contributions	Applicable Gaps
Barbosa, PhD Thesis, 2022 [67]	Consideration of safety and risk in 3D disaster response navigation and more Risk-awareness can be defined in short through a robotic system's internal states (energy, stability) and external hazards (collision, destabilizing factors)	More realistic scenarios for experimentation Assumptions were made for unbounded inputs to dynamic systems and perfect state estimation of system and dynamic obstacle An investigation is needed for sampling-based planners in dynamic cost spaces, that go further then just considering complex, uncertain cost space-limited investigation for apriori, offline risk
Xiao, PhD Thesis, 2019 [62]	formal reasoning of risk in unstructured environments as the probability of the robot not being able to finish the path and a risk-aware planning to minimize the risk objective through a single numerical metric	improvement of viewpoint or teleoperation performance through aerial visual assistance sensor sensitivity to the risk metrics
Choi, PhD Thesis, 2016 [42]	Two-layer collision-free obstacle avoidance algorithm that includes global- and local- path optimization structure generates more energy efficient avoidance trajectory when facing multiple obstacles Development of aircraft controller, obstacle avoidance algorithm in the guidance, navigation, and control, and realistic urban modeling area.	Probabilistic sensor models for sensor fusion System and sensor capability sensitivity to vehicle maneuverability and sensing, Different urban environment explorations modify requirements Safety impacts the desired flight paths in complex urban environments Wind gusts can cause major disturbance to flight stability or maneuverability More realistic urban operation require moving and fixed obstacles
Donato, PhD Thesis, 2017 [37]	Combination of no-thrust airplane point mass model and dubins reachable set theory Steady-state adaptive flight planner with reachable set theory and trimmed flight states GIS algorithms to identify candidate landing sites and estimate risk for people, property, and aircraft Mobile phone activity for population density with temporal dependence	Compensation for any environmental disturbances and modeling errors Selection of an optimal solution for this trade-off between path following feasibility and path planning complexity Safety as a metric for best path must be further investigated since not unique
Fan, PhD Thesis, 2021 [63]	Safety guarantees and accurately quantifying risk through deep neural networks in planning and control Demonstrated an approach to guaranteeing safety while enabling adaptation and learning using Bayesian neural networks. Developed a risk-aware traversability and kinodynamically feasible planning by quantifying and efficiently optimizing over CVaR tail-risks	Consider unknown, non-invertible control gains, or non-control affine systems Transforming LiDAR point clouds to images directly for costmap creation Further application to natural disasters or industrial accidents
Scherer, PhD Thesis, 2011 [35]	Efficient calculations of obstacle cost functions using a novel limited incremental distance transform algorithm Landing site evaluation using a geometric model-based algorithm for VTOL Multiple object motion planning through dynamically feasible routes and multiple objectives	More efficient data structures and algorithms for 3D planning over 100's of km Planning algorithms that consider abnormal scenarios like engine failure Wind and turbulence impacts on flight stability at low-altitude Consideration of other vehicles and people around landing site Pose estimation and environmental sensing sensitivity to system and sensor performance, specifically with range sensors Loosen assumptions of collision risk depending purely on distance to obstacles Land cover and other learned features need subjecting mapping to risk that requires external decisions other than purely geometrical data Improving of initial guess paths and the sensitivity to the global optimal path
Choudhury, PhD Thesis, 2018 [64]	Designing effective planning modules with learning for black-box meta-planners Synthesizing planning policies and collision check algorithms with good finite-time performance Adaptive planning for partially known worlds, kinodynamic planning in high-dimensional space, and real-time constraints with limited computational budget	Learning or initializing good roadmaps to form sparse paths to choose from Improvements in sampling process for incremental densification, so that a policy learns a tradeoff between likelihood of path and quality of objective Consideration of paths as feature extractors where information can be formed from explicitly chosen paths
Ochoa, PhD Thesis, 2021 [49]	Urban flight planning with fail-safe paths Examination of diverse cost metrics with Monte Carlo simulation Comparisons of heuristic and sampling-based planners	Advancement of active perception from new architectures like GANs Additional research is needed to incorporate risk metrics for urban flight planning risk metrics including system, actuator, sensor, and weather-related risks current fixed-altitude maps need to be expanded to full 3D cost maps to support full 3D flight planning
Faigl, PhD Thesis, 2010 [68]	Multi-goal path planning for search missions Formalization of the inspection task with visibility range limits	An extension to a 3D model of environment complex kinodynamic constraints uncertainty in planning sensing and motion costs in the viewpoint planning problem

work focuses on offline flight planning and how to best model uncertainties and prepare for the worst-case scenarios with the available data. The algorithms are combined in a modular framework encompassing a modeling and simulation environment for UAS in disaster scenarios. An example flight plan and outputs for the final demonstration include a flight plan that minimizes risks and can complete flight path objectives as seen in Figure 2.2. Time, risk, and energy metrics are tracked along a flight to measure safety and efficiency while the flight passes through zones for data collection in an urban map.

Research Plan

Develop a framework and software environment for functional and safe aerial disaster response that is modular and flexible to perform Monte Carlo simulations.

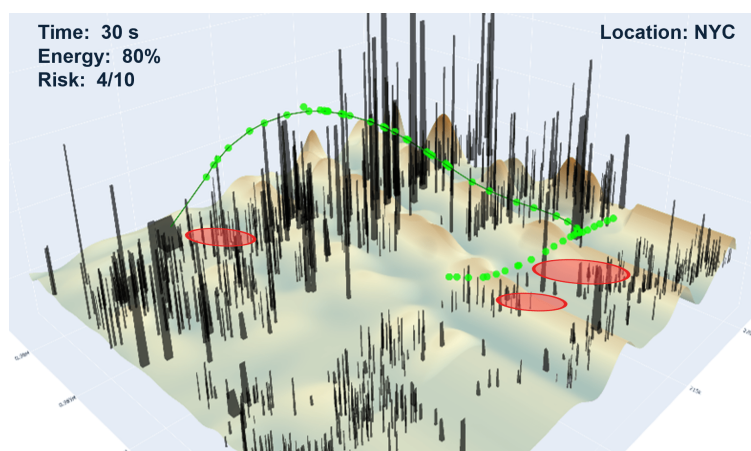


Figure 2.2: Final Implementation Concept

The thesis is structured as follows. Chapter 3 investigates the best approaches to rapidly produce a 3D informative urban map for solving optimal paths. The appropriate data and algorithms are compared for time and accuracy. Machine learning methods are leveraged when the necessary data is unavailable. Chapter 4 explores the best trajectory planning algorithms to find efficient algorithms capable of exploring 3D paths through the urban maps. Algorithms are systematically evaluated to determine which one meets the requirements for this application. Chapter 5 examines the planning problem using a formal reasoning of *risk*, a function that maps to the probability of mission failure. A risk-aware planning algorithm

is developed to explore the urban map with the primary objective of maximizing the probability of mission success. Chapter 6 explores this planning technique for specific disaster scenarios of interest and evaluates the methods from the three previous chapters against benchmarks. Chapter 7 summarizes the results and presents conclusions while including future work ideas.

RAPID URBAN MAPPING WITH LIMITED DATA

3.1 Introduction

The SAFER framework enables offline aerial flight planning for disaster response through a modeling and simulation environment. The environment must be able to represent aerial system planning accurately for the area of the interest, so that it accounts for the constraints and objectives inherent to the planning problem in real-time deployment. Therefore, an accurate map of the environment must be formed quickly during the offline planning task, which will likely be in the midst of disaster response. An increased level of information is needed to make informed decisions while operating under computational constraints. In other words, first responders leverage external sensing to decide where to go and how to get there during uncertain and chaotic situations. The urban disaster response problem requires large-scale urban maps to plan where responders and aerial sensing units should deploy. The offline model of the environment suffers from data uncertainty and availability that makes the problem even more difficult. As an example of simplifying the problem, Kim and Atkins in [73] used a reduced vertex map of an urban environment with a visibility graph for planning flight paths. The compression of the map reduced the accuracy of the optimal paths since the visibility graph is not as accurate as a full graph in the dimensionality of the guidance system. Similarly, Choi in [74] formed a Voronoi grid in an urban map for a UAS delivery system to solve a reduced order version of the planning problem. Each of these methods required the existence of an urban map to find feasible, collision-free flight paths. Current literature lacks an investigation for the best map models when computational resources are limited, and data is not easily accessible. Furthermore, there is no investigation into supplementary methods for mapping 3D maps and environment

features when data is missing and limited. This chapter’s primary concern is the problem of the *Rapid Urban Map*, RUM, where a map for planning is generated quickly for use in real-time or in offline Monte Carlo simulation environments. The following chapters use the rapid urban map to plan and evaluate map metrics. Therefore, the map model must be capable of a few functions including collision checking, feature labeling, and resource tracking.

The strategy used for this work is to separate the terrain, structures, and other obstacles into individual layers for data processing and auxiliary data collection. The modular framework allows each layer to be processed individually using the best algorithm and available data as shown in the next sections. Each feature is combined into the RUM data structure and aligned into a local coordinate reference system with units of meters to allow for planning metrics and measurements to be stored and evaluated quickly. A custom map structure is formed from the independent layers and efficiently stored, queried, and visualized.

3.2 Background

A model of the environment is stored in a custom map data structure to efficiently identify collision-free trajectories, predict trajectory quality using urban metrics, and recognize important objects in the environment. The model starts with a geometric decomposition of the urban environment, primarily from obstacles such as buildings. Various methods for spatially dividing the data and constructing it into geometric primitives like lines and polygons are found in [34]. Previous literature has investigated efficient and informative methods for 3D and geospatial data in the past. However, the methods vary widely in the data types and sources, and the best approach for 3D trajectory planning lacks a detailed investigation. There exists simulation environments and map tools specifically for visualization, construction or climate modeling, and autonomous vehicle research. Each option is often selected based on the accuracy, cost, and ease-of-use for the stakeholders of interest.

3.2.1 Urban Map Modeling

Figure 3.1 shows previous works demonstrating the breadth of literature on this topic. On the left, Harris et al. leverage a Digital Elevation Model (DEM) and high-resolution aerial imagery to predict safe landing zone clusters [75]. On the right, Kim et al. sourced building data in the Miami area to create obstruction maps for UAM flight planning and landing at different altitudes. In another work, Primatesta et al. form binary maps for planning safe flights over urban environments using elevation and population data [56]. In addition, Choi processed LiDAR datasets for large cities and divided the data into building clusters using resampling, and principal component analysis [42].

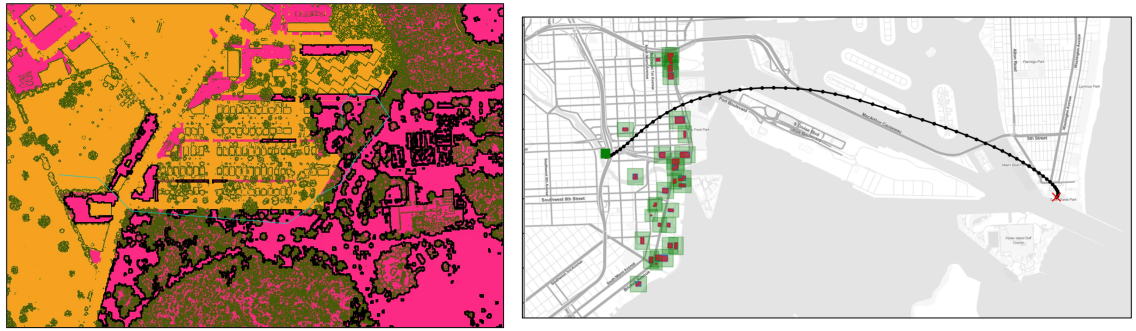


Figure 3.1: Previous demonstrations of Urban Map Models from the author’s previous works in [75] and [46]

Previous work by Harris et al. in [48] investigated the urban map modeling problem specifically for aerial vehicle emergency landing. Geospatial datasets for population density were used to represent the number of people in an area, while structure information from OpenStreetMap, OSM, was used for 3D obstacles. The data was sliced at a single height and combined to use as a costmap for planning. The methodology aimed to estimate obstacles at a single height while providing critical information about the safety of landing at a specific point on the ground in a safe cluster polygon. Therefore, improvements are needed for the disaster response scenario. Figure 3.1 shows how Seulki and Harris used similar geospatial data processing, but focused on finding the occupancy of an urban area at

a specific altitude. A Mixed-Integer-Linear-Programming algorithm solved for the optimal path using a linear dynamics model and time step constraints.

3.2.2 Geospatial Information Systems

Data for large-scale mapping requires geospatial references to match external datasets and visualize features together. Matching features to one another requires the definition of a coordinate reference system, CRS. The EPSG Geodetic Parameter Dataset or EPSG registry provides lists and specifications of different geodetic coordinate systems, from units of degrees to meters. The most well-known geodetic coordinate system is EPSG:4326 or WGS 84, which is used by the Global Positioning System, GPS. WGS 84 is defined by latitude and longitude coordinates in units of degrees. The conversion of georeferenced datasets from WGS 84 to a local meter-based coordinate reference system depends on the area of interest since it is a projection to Euclidean space.

In the United States, a large amount of GIS data is available from regional data hubs. City, county, or state GIS teams across the country have created easy-to-use sites to access extensive data about the city. However, the data availability is limited by the region of interest. The data structures, such as shapefile or GeoTIFF, and the data acquisition methods, like sensor reading or hand labeling, affect the information available for map formation. In addition, data samples' accuracy and uncertainty differ for each dataset.

3.3 Research Question 1

Consider the trivial solution where a digital elevation model, DEM, exists and details the terrain with high-precision georeferenced coordinates of ground height. In addition, all structures exist in a georeferenced dataset. Resampling methods that use nearest-neighbors or average functions on the local data can modify the data to any desired resolution, though the accuracy may be changed. An example of this is seen for Los Angeles County terrain data in Figure 3.2, where the color variation indicates a unique data sample that is later used

for collision checking and feature projection. The memory and spatial search constraints become a concern for a large area of an urban environment with many features. A search of the grid for a specific location will result in a linear $\mathcal{O}(n)$ time complexity, or preferably a log $\mathcal{O}(\log(n))$ time complexity with the use of binary heap. However, by increasing the area of a map by 2, the number of cells has increased by $2^2 = 4$. For example, if the area of interest is 100 km^2 and is discretized into a grid with 10-meter width and length cells, the number of objects, $\{o_1, \dots, o_n\}$, increases to one million. This example of the curse of dimensionality [76] requires careful consideration when building an accurate and informative urban map and promotes alternative methods. The trivial solution that directly combines the DEM and structure data in a discretized grid does not scale well and must be improved.

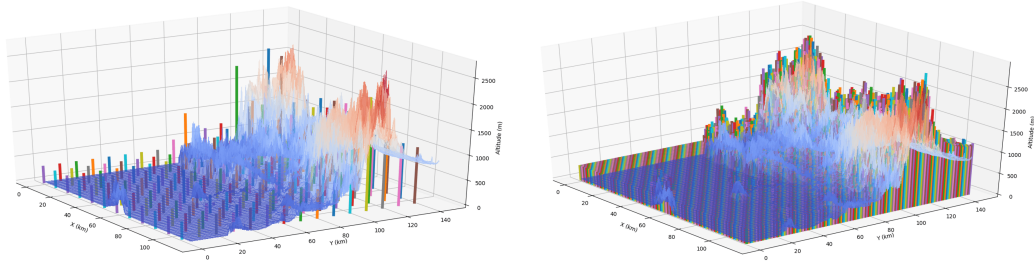


Figure 3.2: Digital Elevation model (DEM) and structure height data for 3D height map. Left side: Grid resolution reduced 10x through resampling; Right side: Grid resolution for 1 km cells

Previous research has investigated the efficient mapping problem through geometric modeling and data clustering algorithms. However, the foundational question remains for this problem.

Research Question 1

What data structures and machine learning algorithms successfully generate urban map models with open-source data and tools for disaster response with field robotics?

The hypothesis is that the urban map model can be formed by independently examining the most critical layers and combining them into a custom software structure that allows

for flexible examination and visualization. Furthermore, the hypothesis is also that supplementary data can be used to fill in the gaps in open-source datasets. Efficient data structures and modern machine learning algorithms can be examined within a single modeling and simulation framework to benchmark and validate the urban mapping performance.

Careful consideration of existing and new data processing algorithms provides insight into an efficient and flexible framework for rapid urban mapping. Previous works show that many geospatial algorithms can efficiently form 3D urban maps if the tradeoff of spatial accuracy and computational complexity are balanced appropriately. Therefore, this chapter investigates whether these algorithms produce maps with accuracy within a reasonable amount and compare which algorithms work best for different scenarios. Furthermore, advanced techniques are used to leverage supplementary data to predict environmental features through data-driven methods and are compared to benchmark methods.

Two experiments are required to answer research question 1. First, a 3D urban map representation is needed for future planning algorithms to explore. The map needs to be created rapidly with the available datasets. Efficient storage and querying of the map are needed for the modeling and simulation environment to be applicable to flight planning. Second, the data pipeline must be upgraded to account for missing data when building a rapid urban map. The missing data is addressed using available visual spectrum and elevation raster data and training a convolutional neural network to predict environmental features, including structure locations.

The framework has the location, time, bounds, resolution, GIS data formats, satellite imagery source, and any additional constraints as inputs. The framework should result in a 3D map object available for visualization, collision-checking, and urban label classification at the selected resolution. An outline of the framework for Rapid Urban Mapping, RUM, is seen in Figure 3.3. The data selections, map algorithms, and integration of map layers are detailed in this chapter, while the use of the map in planning is discussed in future chapters.

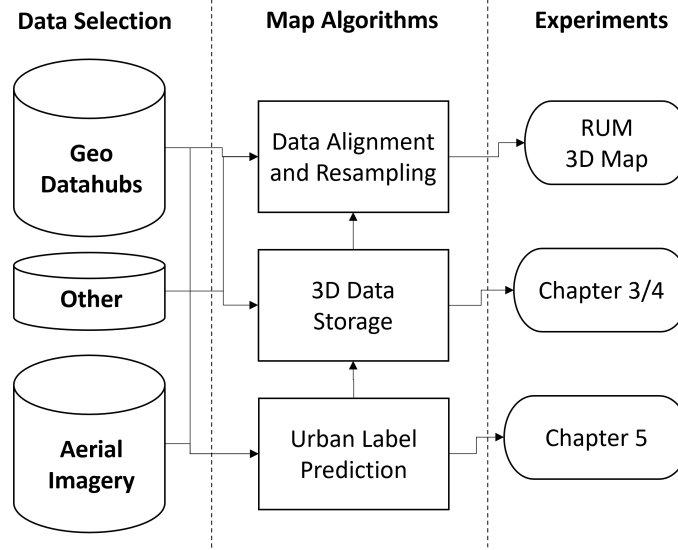


Figure 3.3: RUM Framework

3.4 Urban Map Creation

The rapid urban map, RUM, methodology starts with the selection of data from open-source resources that are useful and accessible. The available data is transformed into a map through four independent layers: terrain, structures, urban features, and weather. The four classes are chosen based on previous literature and are handled independently in order to analyze each layer one at a time. This also allows for a flexible framework as well. More details for each layer are shown in Table 3.1, where the source data, data structures, and other advanced functions are listed from previous literature and experiments. The classes provide information on 3D navigation, local feature items, and how systems operate in the local environment. Today's foundations of the urban map modeling techniques demonstrated in literature are found in the books for modeling the terrain [77] and 3D structural environment [78]. The modeling approach and integration strategy for the layers within the framework are discussed in the following sections.

Table 3.1: Map Data Options

Classification	Source Data	Data Structures	Queries/Functions	Advanced
Terrain	USGS digital elevation model, LiDAR points, USG InSAR	B-spline interpolation, NURBS surface, gaussian process, linear interpolation	Height query collision, Terrain label classification	Offline surface optimization
Buildings	OpenStreetMap, Geo Datahubs, prediction model	Occupancy grid, signed distance field, Polygon primitives, potential field	Collision check, Feature label classification	Elevation matching Storage approximation
Infrastructure and Urban Features	OpenStreetMap, Geo Datahubs			Feature labels for disaster response
Weather	3D model and average wind, CFD Simulations, weather station data	Vector field, Gaussian process, potential field	Flight dynamics wind	Gust modeling for urban planning

3.4.1 Terrain Model

The terrain model provides crucial information for knowing where the aerial system cannot navigate, as the terrain limits the flight path’s altitude. However, it also provides the information or verification of where other structures are bound to the Earth. As detailed in the trivial solution in Figure 3.2, a brute force approach can be taken where a digital elevation model is used directly to query the nearest position for elevation information. However, since this model will be used in the mapping and planning stage, the computational time of generating the data and the accuracy are essential to consider. Therefore, models are compared for how they best approximate digital terrain models and are evaluated for computation and accuracy metrics.

Previous approaches for terrain modeling include the Gaussian process, GP, by Choi et al. [79], non-uniform rational basis spline (NURBS) by Choi et al. [80, 42, 74] and other spline or tessellation methods. The approaches vary in performance and have varying theoretical and experimental success. However, they have not been directly compared for a rapid urban map. The choice is made to compare three methods that cover the majority of terrain modeling approaches and vary in complexity. The B-spline, NURBS, and GP models are selected. Each model is implemented using available Python packages and open-source data. B-spline and Gaussian process models are implemented from the Scipy package [81], and the NURBS model is implemented with the NURBS Python package [82].

Experiment Overview

Experiments are conducted on a 983 squared kilometer region in LA county with sea-level terrain and rolling mountains in close vicinity. The terrain data is from a USGS Digital Elevation Model, DEM, that is reduced to a rectangular region of interest and transformed into a local meter-based coordinate reference system. Parameter optimization is conducted on each method to minimize the residual to a 1-meter resolution DEM before comparing the models to each other. Data resampling in the Google Earth Engine tool uses a nearest neighbors approach and is used for downsampling 1-meter data to lower resolutions. All experiments are run using a Windows laptop with an Intel(R) Core(TM) i7-7700HQ CPU @ 2.80GHz.

B-Spline

The B-spline surface construction is defined in coordinates (u, v) with basis functions $(N_{i,p}(u), N_{j,q}(v))$. The variables p and q are the degree of the surface in the u and v direction, respectively. Algorithms to construct B-spline surfaces select a set of control points, $p_{i,j}$ and knot vectors, U, V , resulting in a surface in homogenous coordinates defined by the linear combination of the B-spline basis functions.

$$p(u, v) = \sum_{i=0}^m \sum_{j=0}^n N_{i,p}(u) N_{j,q}(v) p_{i,j} \quad (3.1)$$

The surface of a B-spline can be fit using a set of data points. Since the B-spline is a linear combination of basis functions, the surface can be solved using least squares minimization. There are two metrics of interest when interpolating data with a surface model. One is the difference between the predicted value and the observed value, or residual. The other is the difference between the predicted value and the true value, or error. A visualization for 1D data is shown in Figure 3.4. The residual can tell how well a model is trained to the observed data, but the error provides insight into how well the model generalizes to

the true shape of the terrain.

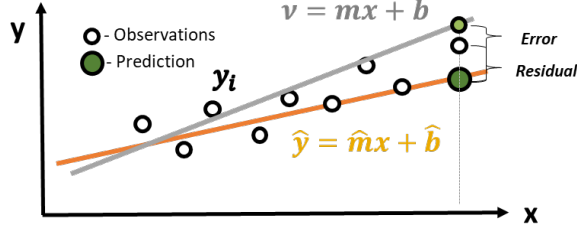


Figure 3.4: Example for Residual and Error

A tradeoff exists between a model's computational time and accuracy for rapid urban mapping. In Figure 3.5, the computational time and residual per sample are shown for different resolutions of the training datasets. The DEM model is a 1-meter resolution dataset that is resampled to lower resolutions to reduce the computational overhead and evaluate how the model's performance changes depending on the input size. Figure 3.5 and the respective figures for the other terrain models show columns for the time to train the model in blue and to query the model for the estimated height, \hat{z} , in yellow. The line plot shows the per sample residual in meters, which ranges from zero to over 20 meters for the B-spline.

$$\hat{z} = model(x, y) \quad (3.2)$$

The model can also be evaluated against the original 1-meter DEM model to find the total error. The Mean Squared Error in Equation 3.3 is calculated using the 10-meter B-spline model and results in an error per sample of 20.2 meters. The MSE is compared against the other two models to determine the model that performs best at 10-meter resolution.

$$MSE = \sum_i (\hat{y} - y)^2 \quad (3.3)$$

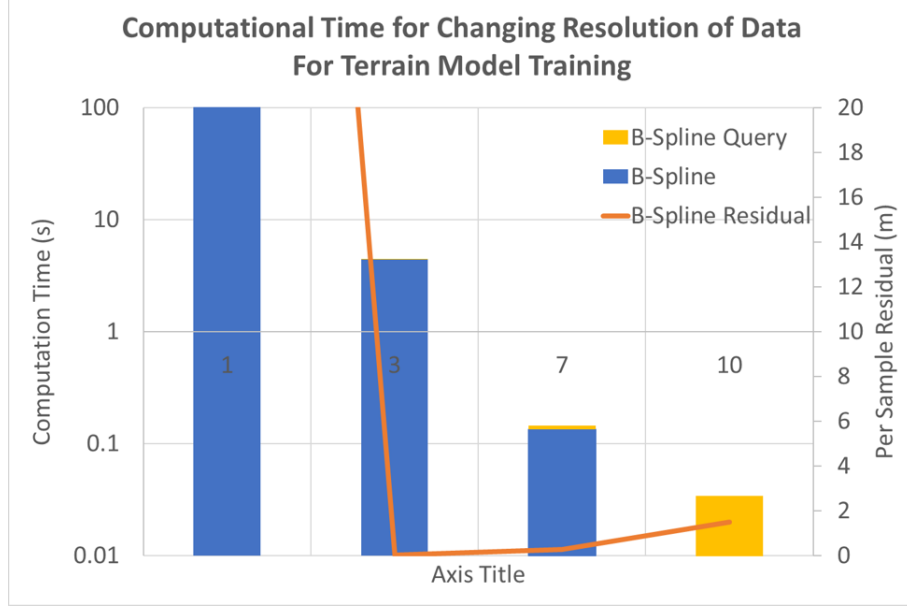


Figure 3.5: Computation Time and Residual for B-Spline Training

NURBS

An extension of the B-spline using homogeneous coordinates is the non-uniform rational B-spline, NURBS. The surface form differs from B-splines by weighting the control points, $w_{i,j}$. The equations defining the B-spline curves are in Equation 3.4 and Equation 3.5.

$$s(u, v) = \sum_{i=0}^k \sum_{j=0}^t R_{i,j}(u, v) p_{i,j} \quad (3.4)$$

$$R_{i,j}(u, v) = \frac{N_{i,p}(u) N_{j,m}(v) w_{i,j}}{\sum_{p=1}^k \sum_{q=0}^t N_{p,n}(u) N_{q,m}(v) w_{p,q}} \quad (3.5)$$

An iterative solution to the surface fitting follows a sequence of knot insertion, knot removal, and degree elevation to find the surface with minimal residual. The NURBS model has been used for terrain modeling successfully and is detailed in [83, 79]. The training results are seen in Figure 3.6, where the residuals and computational time for fitting the surface are very similar to B-spline. However, while the residual is reduced by about 1-meter, the computational time to query the surface for the height at a position is

significantly higher. Furthermore, at the 10-meter resolution, the training time has now become significantly greater, though it is still small.

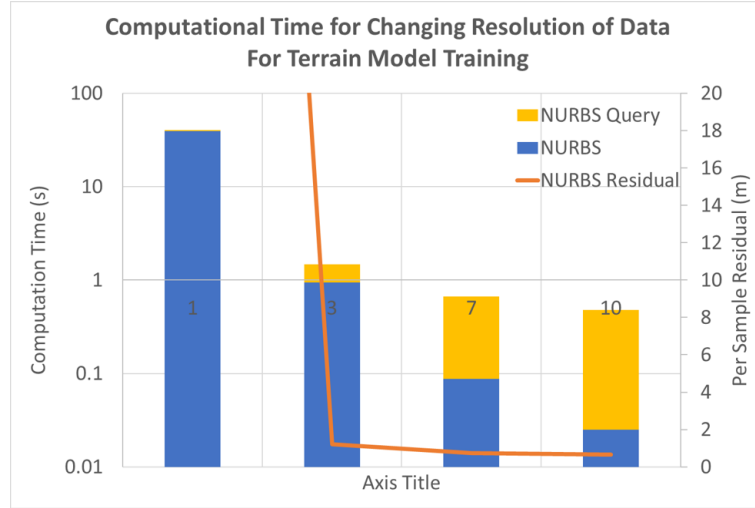


Figure 3.6: Computation Time and Residual for NURBS Training

While the residual and time are significantly higher for NURBS at a 10-meter resolution, the error per sample is lower at 16.93 meters, which is a 16% improvement on B-spline. This fact confirms the assumptions that the more complex NURBS model can better fit the terrain in mountainous areas, where the terrain changes are nonlinear and multimodal. However, as shown by the comparison of NURBS and B-spline, the more complex the model, then the more parameters and training are required.

Gaussian Process

Gaussian processes (GPs) are non-parametric models that can fit data using Gaussian probability distribution in the function space. The model is defined by a unique mean and covariance function for the surface fit of interest. Gaussian processes are often used for regression because of the ability to find the best fit over the distribution of all possible fits for the data. The GP is a discriminative model, not a generative model, but evaluates the data points over a distribution of functions.

Table 3.2: Radial Basis Function Parameters

Noise Std	Length Scale	Length Bounds
5	15	[1e-1, 1e3]

$$G(X) \sim GP(m(X), k(X, X')) \quad (3.6)$$

The mean function, $m(X)$ and the covariance function, $k(X, X')$ represent a distribution over functions [84]. The covariance function, otherwise known as the kernel function, models the relationships between the data points. The commonly used Radial Basis Function, RBF, kernel equation is seen in Equation 3.7. The GP training algorithm involves an optimization problem where the design variables are the hyperparameters of the kernel function, and the objective function is the maximum marginal likelihood estimation function. The GP model was used for terrain modeling in [79, 84].

$$k(x, x') = e^{-\frac{\|x-x'\|^2}{2\sigma^2}} \quad (3.7)$$

The RBF kernel is selected for the GP model because of the previous success and the few parameters required. The parameters are compared over a set of 30 tests, and the best parameters are seen in Table 3.2. Since the GP training is computationally expensive, a separation parameter is included that skips every n data points. Training time is then kept closer to that of B-spline or NURBS, using the same resolution for the dataset. In addition, a constant kernel is used to scale the RBF kernel depending on the terrain data type.

The Gaussian Process differs from the other two models in a unique way. The residual continues decreasing as the resolution gets very small, for example, from 10-meter to 1-meters. As expected, a more complex model improves in accuracy as the data resolution and dataset size increase. The concern is that the computational time to train the model is significantly higher, with up to 100% more time. However, as detailed earlier, the GP model provides not an interpolated height value but a mean and variance for the output.

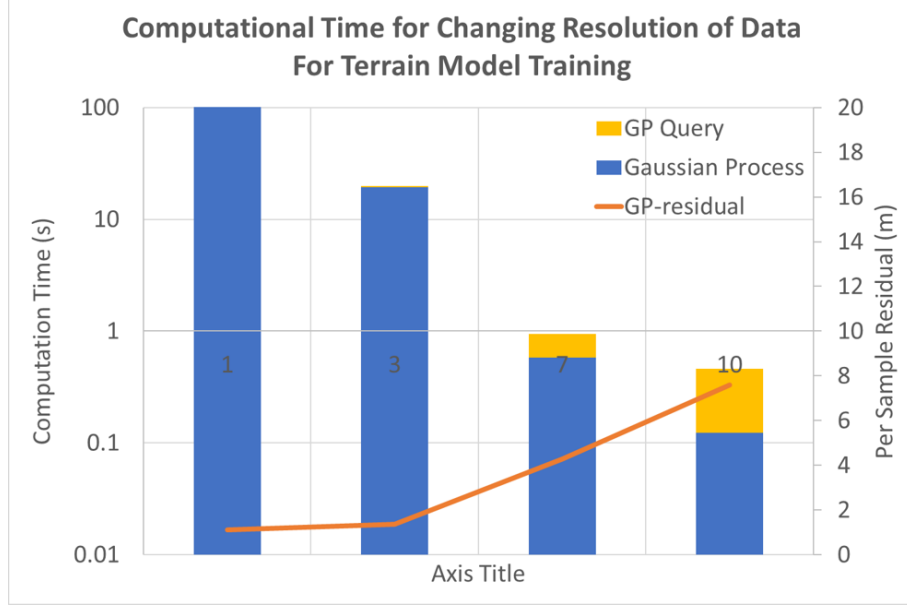


Figure 3.7: Computation Time and Residual for GP Training

The total error for the GP model at a 10-meter resolution is 16.97, which is approximately equal to the NURBS error. However, as shown earlier, the GP error improves by reducing the training dataset resolution, as shown in Figure 3.7, and therefore outperforms the NURBS model. Furthermore, the GP model outputs not only a prediction value but also the variance of the prediction, as shown in Equation 3.8. A visualization of the performance compared to the other two models is shown in Figure 3.8. The comparison between the three models is discussed in the next section.

$$\mu_z, \sigma_z^2 = GP(x, y) \quad (3.8)$$

Model Comparison

A 2D slice of the terrain model predictions is shown in Figure 3.9. The shape is generally tracked by all three of the terrain models, but each model differs in accuracy and consistency. The B-spline model tends to track training points too tightly, resulting in poor predictions at more extreme jumps in terrain. The NURBS model lags behind the changes,

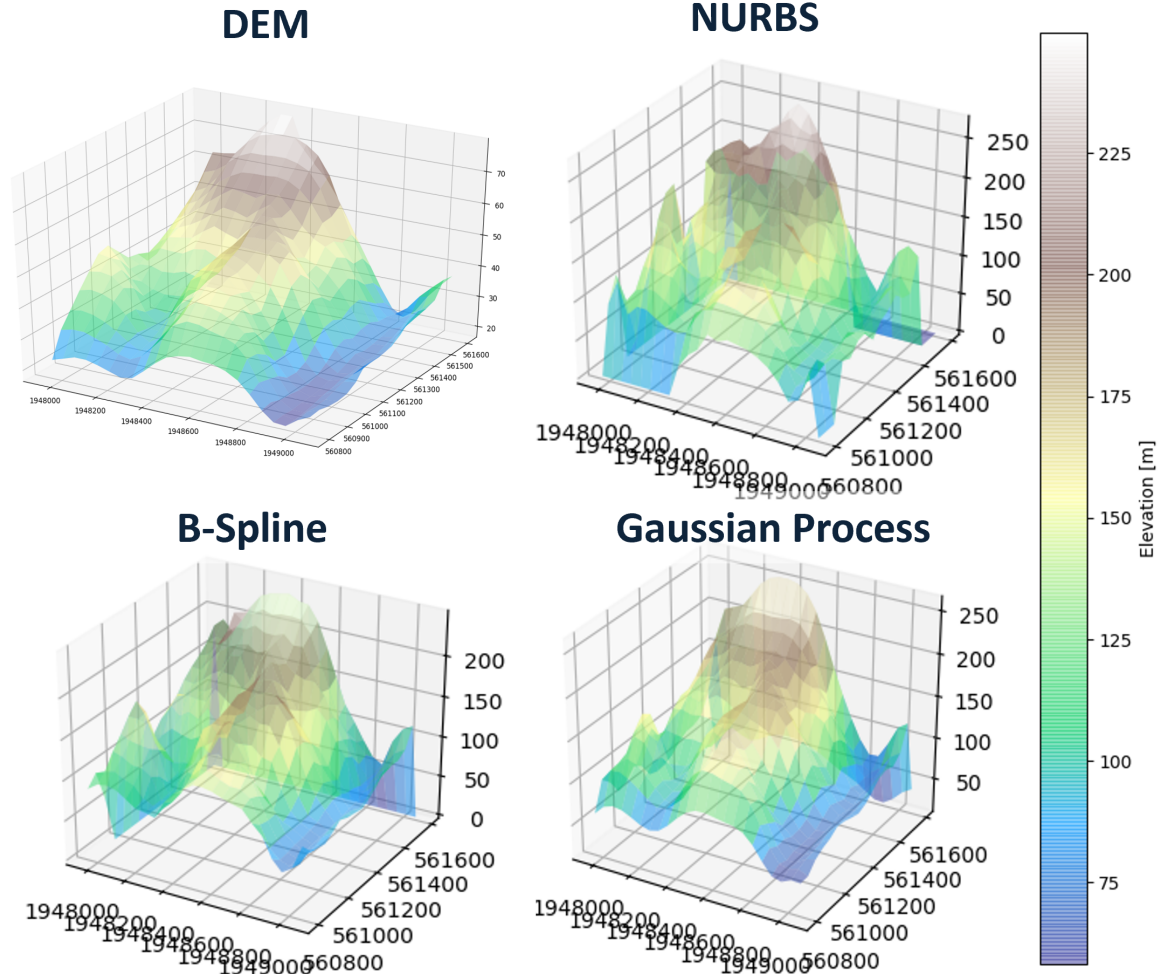


Figure 3.8: Terrain Model Surface Visualizations on LA

resulting in a consistent error. However, the general shape of the environment is tracked well, as shown in Figure 3.8. The average error per sample from two hundred thousand data points at 1-meter resolution was 79, 42, and 32 for B-spline, NURBS, and GP, respectively. The comparison of the models at a 10-meter resolution is shown in Table 3.3. While the B-spline algorithm is much faster to compute, it results in a lower accuracy model, as may be expected by its simplicity compared to the other two methods. The NURBS model shows slightly better accuracy in this test case while computing significantly faster. However, the Gaussian Process model's direct modeling of the uncertainty through a probability distribution makes it powerful. For this reason, the GP model is selected to use moving forward

Table 3.3: Terrain Model Comparison

Resolution: 10 m (10x)	B-Spline	NURBS	GP	DEM
Error per sample to Truth @ 1m	79.27	41.92	32.73	0
Residual per sample in training	0.35	4.10	1.62	0
Computational Time to Train (s)	32.63	104.10	141.99	60.0 (load)
Query Time (s) (per 1000)	0.011	0.58	0.36	1.43 * 1000
Bayesian Outputs	Sample residual	Sample residual	σ^2 in Z	10 cm RMSE
Storage Size (kilobytes)	161	191	783	54,793

for terrain modeling.

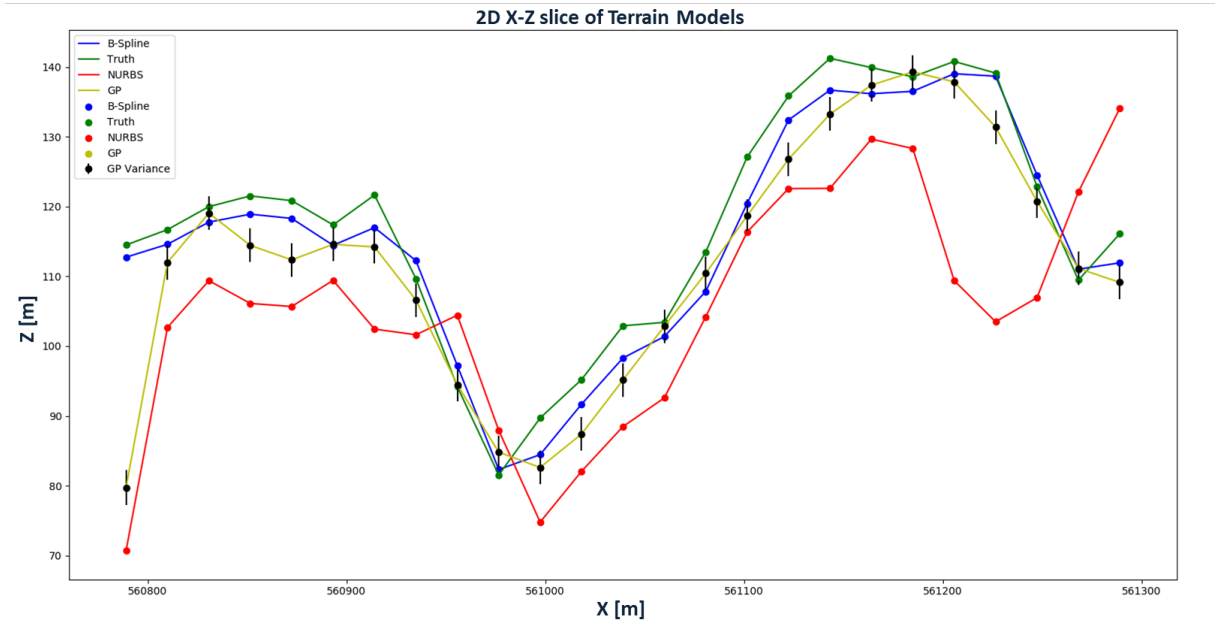


Figure 3.9: 2D Slice of Terrain Models in X-Z

3.4.2 Structures

The other primary risk of collision or danger in the flight path is 3D structures and buildings. The positions of buildings in most urban metros have been mapped through community sources like OpenStreetMap, but the data is often incomplete and lacks high-quality standards. Some urban metros have Geo datahub sources with validated structure maps that have quality standards for upkeep and distribution. The datasets are often provided in Shapefile formats, which represent each structure with a polygon. The polygon approximates the shape of the building in the coordinate reference system selected by projecting

the geometric object. A data pipeline is created to process these structure models and map them onto the terrain model.

The 3D extrusion of the structures requires additional data. An assumption is made that the structures extend in the z-axis together, even though many structures have angled or tiered roofs. The elevation of the z-coordinate of each building is found by querying the terrain model in Equation 3.9. The extension in the z-axis is found by using the height feature from the structure dataset, D , assuming the data is available as shown in Equation 3.10.

$$\hat{z}_0 = \tau(x, y) \quad (3.9)$$

$$\hat{z}_i = \tau(x, y) + D(i, 'height') \quad (3.10)$$

Building shapes are approximated to rectangular objects and then stored by the maximum latitude and longitude bounds. This can cause distortion to the polygon and add an undefined buffer around each obstacle because of the orientation of the coordinate system transformations and the principal axes. An improvement is made by aligning the structures with the local coordinate axes and discretizing them into the desired resolution, as shown in Figure 3.10. The height parameter, h , is used to scale the cell matching to the matching structure and is constant for the polygon representing the structure. The resolution, r , modifies the number of cells to compute for each structure. The angle, λ , represents the shift from the building's principal axes to align to the grid. The r and λ values are selected from an iteration of the threshold parameters discussed in the next section.

Similar to the terrain data, when searching for occupied positions in space, the use of an efficient data structure can improve speed and accuracy. In particular, collision checks against buildings repeatedly occur during planning, and the distance to structures and between structures may be solved thousands of times. A commonly used solution to this problem is to leverage a K-d tree data structure or the k-nearest neighbors, KNN, algo-

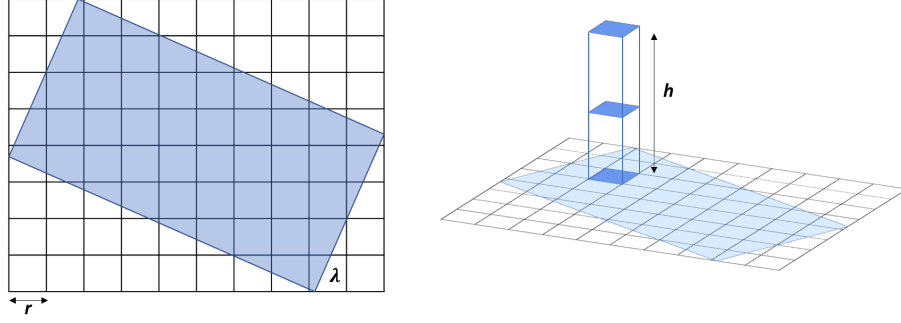


Figure 3.10: Building Discretization and Height Formation

Table 3.4: Comparison of Brute Force and K-Nearest-Neighbors search whether a position is occupied in the map

Buildings for Collision Check	Brute Force	KNN (K=5)
1,358	8.06E-4	1.63E-4
135,800	9.06E-3 (100x)	3.26E-4 (2x)

rithm. This reduces the search to find the nearest objects from $\mathcal{O}(k * n)$ to $\mathcal{O}(k * \log(n))$. In an experiment in the Manhattan area of New York City, a subset of 1,358 structures demonstrated the speed increase between the brute force search and KNN approaches. The results shown in Table 3.4 confirm the theoretical improvement since $\log(100) = 2$.

Structure Data Discretization

The structure representation can be approximated or reduced to improve speed and reduce memory for searching for map obstructions. For example, consider the basic form of the algorithm for searching for collisions along a given path in 3D space. First, each 3D position, x_i , is considered for a path with s positions and must find the closest of n obstacles using a binary search. Then, the closest obstacle, or obstacles, o_k are checked if they occupy the space of the state x_i . This results in a time complexity of $\mathcal{O}(s * \log(n) * c)$. Other than reducing the points or number of structures, which is a function of the planner and map, the only other improvement is to speed up the time to check whether an obstacle causes a collision, c . In other words, this is the time to check if position x_i is occupied by an obstacle o_k . Therefore, if the obstacles are discretized too small, the number of checks becomes

Table 3.5: Structure Discretization Experiment using Washington DC Boundary Zone

<i>Inputs</i>		<i>Outputs</i>		
Side Length (km)	Threshold	Count	Area Ratio	Overlap Ratio
1.0	0.90	152	0.851	0.849
2.0	0.60	42	0.957	0.892
0.5	0.95	676	0.922	0.921

a bottleneck in the algorithm. A demonstration of the shape discretization using the Washington DC boundary is detailed in Table 3.5 and Figure 3.11. Parameter selection is critical to getting the correct accuracy while not requiring large amounts of cells. In the scale of kilometers, the DC boundary can be filled up to about 85% with minimal overlap using 1 km length grid cells. These results are preferred compared to the additional area and large amount of grid cells of the alternative input selections. However, alternative methods to buffer the grid and align the data is available with external tools. Therefore, the Feature Manipulation Engine, FME, Workbench tool [85] is used in the next section for verification and examination of the structure mapping method.

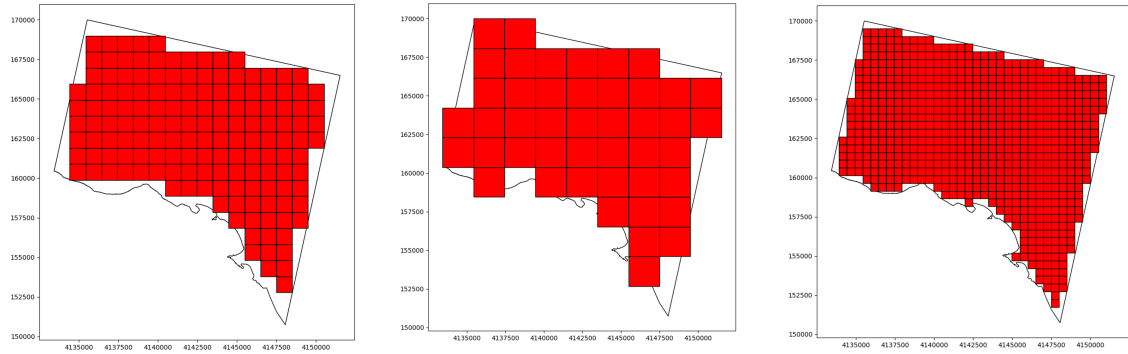


Figure 3.11: Structure Discretization Experiment using Washington DC Boundary Zone

Verification and Validation

The FME Workbench is a model-based, graphical tool for analyzing 2D or 3D geospatial data. The tool compiles data together in a graph and passes information through functions for sorting and visualizing data based on 3D spatial data, class features, and metadata. The pipeline for using FME is shown in Figure 3.35. A series of building approximation meth-

ods are implemented and evaluated in FME to compare the accuracy and time for each method while verifying the models and validating existing datasets. The validation data is the cityGML model of New York City available from the City of New York Geo Datahub. The results are shown in Figure 3.12, where the clear indication is that a bounding box method increases the total volume of the map, resulting in a buffer around obstacles. The volume error can be contained within 15% by using a 10-meter discretization. This varies from the previous section, as the FME tool sets a low threshold value and maximizes the coverage of the structure, causing a large buffer around the 3D object. The computational time reduces significantly if a large discretization is used, for example, 20-meters. Examples of this through images is seen in Figure 3.13 and Figure 3.14. In addition, the volume error and calculation time between the use of a bounding box and the rotated polygon of the building objects is minimal and not a concern. The labels with 'BB' indicate that a rectangular bounding box was placed around the structure with the shape along the global axes. The labels with 'City' indicate that the structures directly from the city dataset were discretized along the direction that best forms the square grid cells.

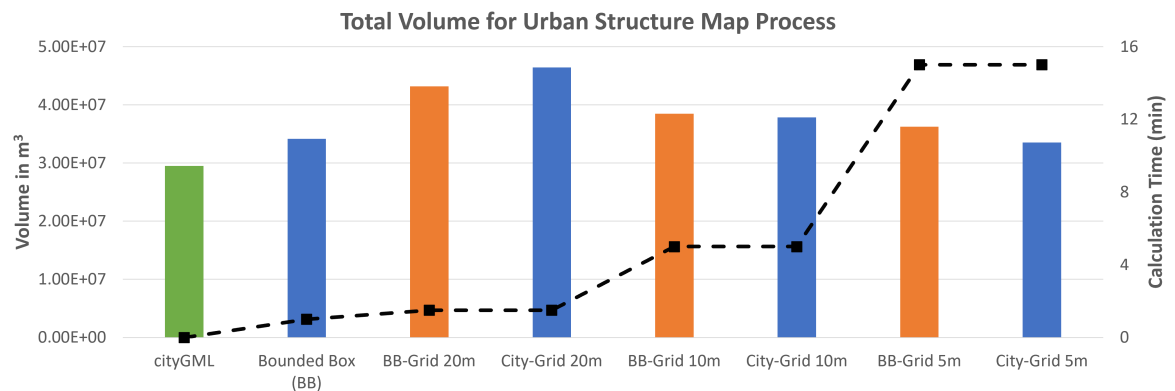


Figure 3.12: Changing Volume and Computational Time for Differential Structure Approximations

Recent advancements in urban modeling include the standards of cityJSON and cityGML, with tools and documentation primarily coming from Delft University of Technology (TU Delft). The standardization and ease of use from available functions are exciting but are

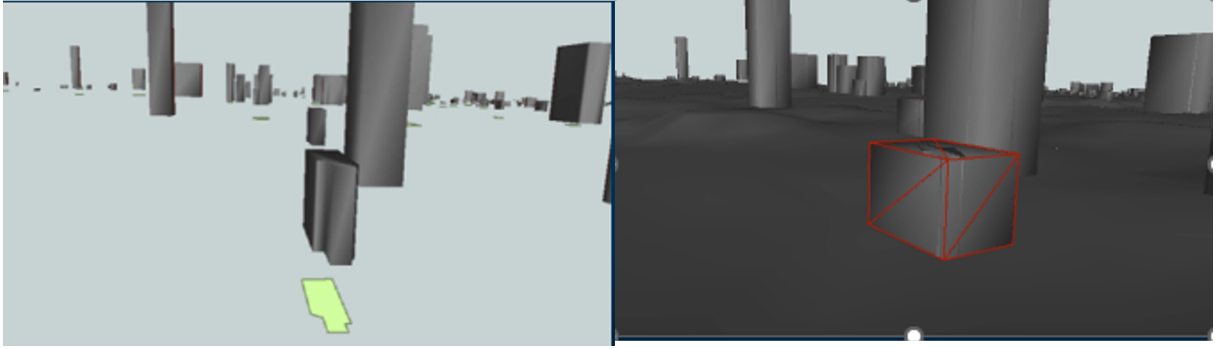


Figure 3.13: Bounding Box Buffer on Buildings and Terrain Validation

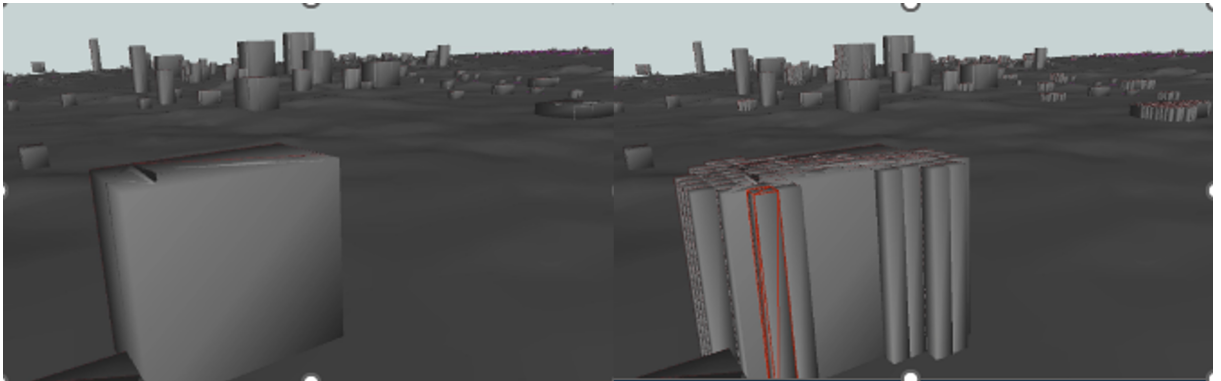


Figure 3.14: Discretization Impact on Buffer

more suited for future work. The current environment suffers from large files and programming inconsistency with this work. For example, the New York City file is over 3 GB in size, and there is no standard python package that can easily preprocess this file. Therefore, the custom RUM framework outperforms the standards for this application. However, the standards should be further investigated in the future and can be a great source for verification and validation.

3.4.3 Weather

Previous explorations or wind mapping for simulated flights have mostly been for flow over terrain without structures or for low-resolution climate models with structures. Urban wind modeling is largely impacted by the buildings and structures, and thus often requires Computational Fluid Dynamic, CFD, simulations to accurately represent the wind speed

and direction in urban canyons and alleyways. Previous literature has successfully demonstrated accurate urban wind maps using these techniques, but the algorithms and data sets are computationally expensive and requires additional complex models to accurately model the atmosphere and flow dynamics.

As a simplifying assumption, the wind in this work is treated as a constant wind field with an added gust model to account for structures and aerial system movement. The Global Wind Atlas¹ is used for accessing data and exploring the temporal nature of wind. The gust is modeled using the Dryden gust model that is affected by structures in the vicinity. The model provides enough insight into smaller aerial systems while avoiding complex urban map CFD runs. The Dryden model is used with an additional noise parameter for the uncertainty from building gusts and weather changes. The Dryden model parameters, L , and σ , seen in Equation 3.11, are computed offline over the map of interest, and the gust model is queried when needed during the flight simulation as a random sampling, shown in Equation 3.12. The total wind vector is the combination of the wind field and the stochastic gust vector.

$$G(s) = \sigma \sqrt{\frac{2L}{\pi V}} \frac{1 + \frac{2\sqrt{3}L}{s}}{(1 + \frac{2L}{V}s)^2} U(s) \quad (3.11)$$

$$\vec{W}_V = \vec{W}_F(x, y) + G(\mathbf{s}) \quad (3.12)$$

3.4.4 Summary and Results

The RUM methodology successfully creates 3D representations of urban environments using open-source datasets and efficient algorithms. The map is divided into four layers including terrain and structures, and open-source data is sourced to rapidly form an accurate map for use in planning algorithms. The RUM maps take between 20 seconds to five minutes to form depending on the area and parameters selected. The B-spline, NURBS,

¹<https://globalwindatlas.info/>

and Gaussian Process models are compared for terrain modeling of a large-scale and mountainous region in Los Angeles County. The Gaussian Process model is selected for having the lowest error at a 10-meter resolution and for the usefulness with Bayesian outputs. The Bayesian outputs will be used in Chapter 5. Structures are formed using open-source data, mainly from Geo datahubs from the cities of interest. The data is optimized for storage and collision checking using geospatial transformations and polygon fitting.

The environment is successfully used to find 3D collision-free paths. A 3D path is found that explores New York City without colliding into structure or terrain models using the RRT algorithm detailed in Chapter 4. The algorithm can explore down city corridors with the blue dots representing nodes and the final path shown in green. More discussion on the planner is featured in Chapter 4.

A summary of the different urban maps created using the RUM pipeline is shown in Figure 3.15. The metropolitan areas of Los Angeles, New York City, and Atlanta are selected to 3D maps capable of visualization and future planning work. For all the maps, a structure discretization size of 20 meters is used, and the DEM is resampled to 10 meters. A validation study is conducted in the Atlanta area. The Google Earth platform is used to visualize geometric primitives, points, and lines. The Earth Engine and FME Workbench are used to verify the software and validate the mapping of data to the RUM model.

The full pipeline summarizing the flow of data is seen in Figure 3.35. As a demonstration of the RUM pipeline, a portion of Atlanta, Georgia, is modeled and verified in the FME software. The data for Atlanta is compiled from OpenStreetMap's structure data and the USGS 3D Elevation Program's 1-meter Digital Elevation Model from OpenTopography². The map is seen in Figure 3.15 and Figure 3.16.

Now that a rapid urban map is formed from structure and terrain data, the question remains of how the same methodology can be achieved when data is missing or when additional feature types of necessary for aerial situational awareness tasks. For example, open

²<https://opentopography.org>


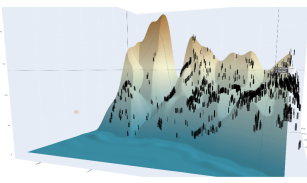
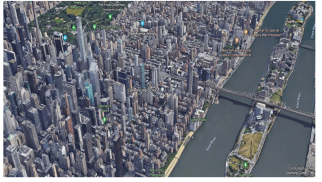
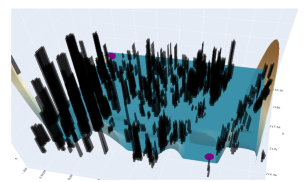

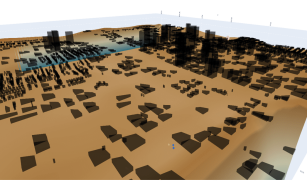
Location	Data Description	Google Earth Image	Rapid Urban Map
<i>Los Angeles, California</i>	Area: 15.38 km ² Structures: 6950 Size: 118 MB		
<i>New York City, New York</i>	Area: 1.91 km ² Structures: 1472 Size: 77 MB		
<i>Atlanta, Georgia</i>	Area: 9.56 km ² Structures: 325 Size: 202 MB		

Figure 3.15: Rapid Urban Map Model Examples

source data is often missing the location of smaller structures or has inaccurate height data for dense areas. Furthermore, the primary objective of situational awareness requires additional insight into the trajectory from land cover features. The following section explores these methods of predicting urban map features to address this limitation.

3.5 Urban Map Updates

The urban map requires knowledge of the environment's structures and other obstacles. While terrain data is mostly consistent year-to-year, the locations of buildings and low-altitude obstacles change, especially during times of economic growth. Therefore, there is a gap in how to build rapid urban maps when there are outdated geospatial datasets. For example, the city of Atlanta has grown rapidly in the past ten years, which has changed both the terrain and building structures, as seen in Figure 3.17. New construction has begun, with some new buildings being built and some buildings being torn down. There is a need for a data-driven technique that can predict urban region labels, specifically structures and

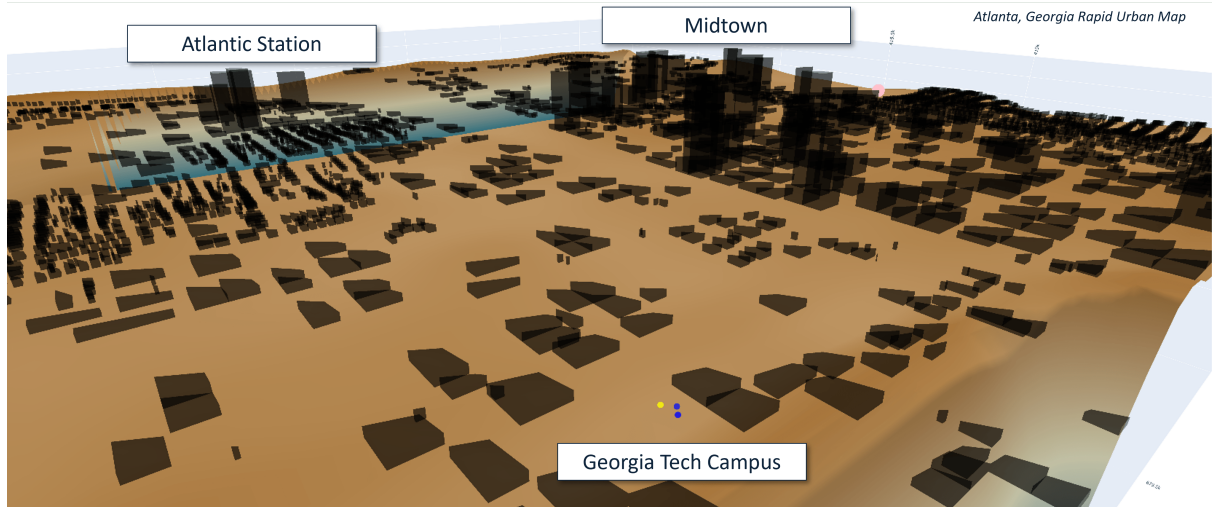


Figure 3.16: Rapid Urban Map Model of Atlanta, Georgia

low-altitude obstacles.

Many companies now rely on updated maps to provide customers with traffic information or simulate navigation for future autonomous vehicles and there is an extensive market for accessing these up-to-date 3D maps. This area of research has grown in recent years with the growth of realistic video games and autonomous vehicle navigation. Companies such as MAXAR³ are now devoted to providing digital twins of cities. These "living cities" or digital twins of urban regions are computationally expensive to create and, therefore, costly to access. The assumption is made that the maps are too expensive for stakeholders, and the level of detail is not necessary for the maps in this work. Furthermore, Monte Carlo simulations require a fast, efficient tool that is not dependent on a large-scale digital twin.

Therefore, an alternative method is needed to supplement the data for the RUM method. Landcover predictions or urban feature labels are needed to define the model. Two public datasets predict the land cover across the United States. The USGS National Landcover Database (NLCD) is a 30-meter resolution dataset that provides landcover classification labels for the United States. The database leverages Landsat⁴ satellite multispectral data, and

³<https://blog.maxar.com/earth-intelligence/2022/digital-twins-power-autonomous-navigation-with-precision3d>

⁴<https://www.usgs.gov/landsat-missions>

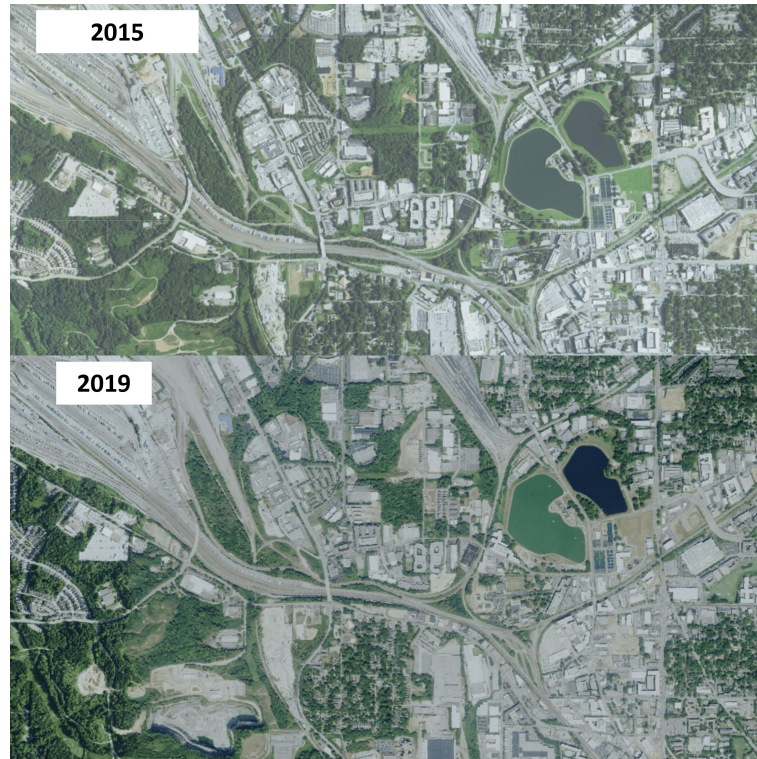


Figure 3.17: Difference in Aerial Views of Atlanta (Westside and Midtown) circa 2015 to 2019 from NAIP imagery

the most recent release is in 2019. The dataset covers 20 different land cover classifications and additional impervious labels. This level of disaggregation is not required for the rapid urban map, though, and the 30-meter resolution fails to precisely detect individual features like buildings. Furthermore, the database is only updated every 2-3 years and therefore is missing changes in the environment, such as in Figure 3.17. The Google Dynamic World landcover prediction, GDW is a 10-meter resolution near-real-time dataset that provides 9 class label probabilities. The predictions are generated from Sentinel-2⁵ L1C images with clouds filtered. The comparison between the labels are available in the Appendix in Figure B.3. The dataset roughly provides the features of interest; however, the urban environment is still defined by the labels of "Built-up area" and "Bare Ground," which do not include precise locations for structures in the environment. The resolution is successful for creating informative maps but is less than desired for producing accurate urban maps for

⁵<https://sentinel.esa.int/web/sentinel/missions/sentinel-2>

planning and visualization. For example, Figure 3.18 shows the differences between the maps of NLCD, GDW, and the custom labels from RUM that will be presented later in the chapter. However, this comparison reveals that the detail and information provided by the two other methods are much less than desired, specifically in an urban environment and for this urban mapping task.

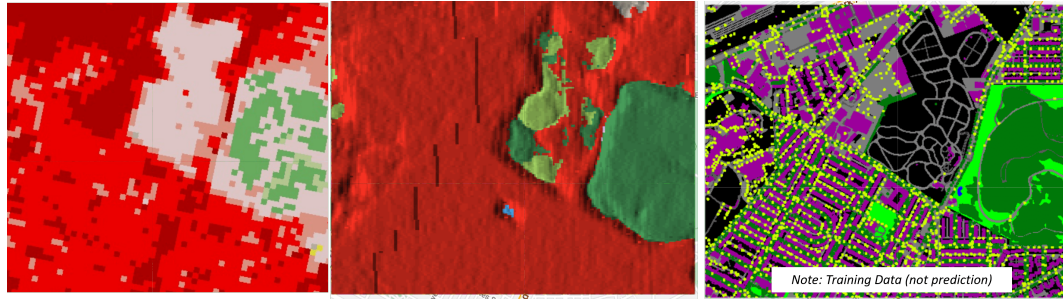


Figure 3.18: Comparison of Labels and Resolution of NLCD, GDW, and RUM Landcover Mapping in Washington D.C.

In the last ten years since the Deep Learning Revolution restarted with ImageNet [86], the field of deep learning for imagery has exploded. The capability of convolutional neural networks was proven in the earlier years of Artificial Intelligence, AI, and the modern computational resources and open-source software packages like Tensorflow and Pytorch have led to rapid growth. Now, any student or researcher with a laptop can deploy and train convolutional neural networks and more. Network architectures continue to improve with models such as ResNet, UNet [87], and Pix2Pix [88]. The broad set of features detected from the visual spectrum data is possible because of the rich features that are both semantic and geometric [89], which eyes see as textures, depth, and color. The increased accessibility of image datasets has improved training during the modern phase of the deep learning revolution. In recent years, Andrew Ng has urged for a shift in focus for data-centric AI⁶, as opposed to the previous model-centric approach. For these reasons, many works have investigated deep learning applications and datasets for learning to predict environmental features, as opposed to model architecture modifications.

⁶<https://landing.ai/data-centric-ai/>

Convolutional neural networks, CNNs, are deep neural networks that leverage a spatial relationship between nearby data samples. One of the key reasons that CNNs are used is because of the use of convolution rather than matrix multiplication in their layers, as detailed by Goodfellow in [90]. This is achieved using kernel multiplications (convolutions), nonlinear activation functions, and pooling stages to form a CNN output. In addition, Back-propagation is used for gradient-based parameter updates to speed up training, especially when using graphical processing units, GPUs.

The assumption is made that the problem can be solved in part by supervised learning. There exists a set of labels, y , that are assigned to data feature samples \tilde{x} , and a CNN model $f(x, w)$, defined by weights, w , is to be trained through a loss function $J(y, \hat{y})$ by gradient-based weight updates.

3.5.1 Learning-based Map Updates with Aerial Imagery

Extensive work has investigated how to develop maps for urban environments with structures and other obstacles. For example, work by Wang et al. in [91] leveraged remote sensing data to accurately characterized urban structures in 3D. In particular, Landsat imagery and elevation data are used to build an urban map. Previous works have demonstrated the ability to detect informative features from the environment using visual information from bird's-eye view data, such as satellite or aerial fly-over images. This data can be used to detect objects [92] or classify the land cover of the environment [93]. It has been used for applications, including road and infrastructure mapping [94], as well as flood mapping [95, 96]. Emergency response applications have used aerial imagery in the past for situational awareness, whether to detect missing persons [97] or to monitor wildfires [98]. Various methods do this, and pattern recognition techniques using machine learning are often discussed as classical computer vision. Modern-day research in this field often leverages deep learning and convolutional neural networks because they are powerful at learning the critical features within a dataset of images. More details are available in **Computer vision**

algorithms and applications [99] by Szeliski.

In recent years, Convolutional Neural Networks have become the standard for these types of problems because of a few important features. CNNs leverage sparse interactions, parameter sharing, and equivariant representations that allow for efficient learning, high-dimensional latent spaces, and spatial relationships between features. Mathematically, the CNN is efficient by using convolution as opposed to matrix multiplication and is robust because the pooling layer promotes approximately invariant features from minor changes to the input.

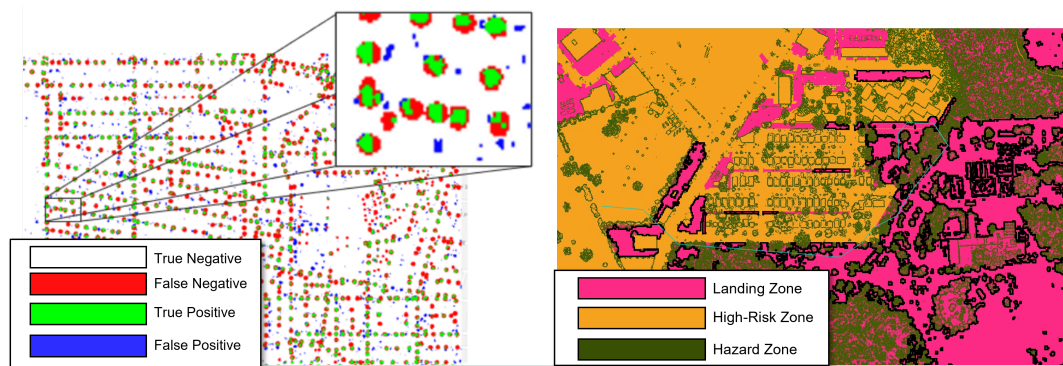


Figure 3.19: Previous Applications of the U-NET Architecture from [100] and [75]

Neupane et al. in [101] performed a review and meta-analysis of deep learning-based semantic segmentation methods for detecting urban features in satellite images. This included an extensive table documenting past frameworks, classes, models, and reported performance metrics. This review formalized the knowledge that Deep Learning performs well with aerial imagery and multi-label classification tasks. The convolutional neural network models range from ResNet to VGG-16 to various takes on U-Net. Previous research leveraged the U-Net architecture for detecting utility poles [100] and for classifying landing zone risk [75] as shown in Figure 3.19. Neupane also introduced the case for leveraging Generative Adversarial Networks, GAN, for high-resolution, multi-class feature learning. One method applied a conditional GAN to Sentinel-2 imagery to classify water, developments, forests, grass, pastures, and cultivated labels [102]. Another example used high-

resolution imagery from monocular satellites to detect the classes trees, buildings, mixture, and pagodas. The network succeeded after optimizing the parameters [103]. In SatGAN, the Pix2Pix cGAN was used, but with modifications to the loss with a perceptual reconstruction loss metric [104]. This can be compared to similar applications where a CNN is used for building, grassland, dense vegetation, waterbody, barren land, road, and shadow; however, the result is dependent on patch size [105], a problem that GANs do not have.

A GAN is a CNN architecture that takes a Bayesian modeling approach and assumes the problem to be an image-to-image translation. The development of GANs is based on the game theoretic idea and features two networks, the generator and discriminator, competing in performance during training [90]. Conditional GANs, cGANs, seek to learn a conditional distribution $p(x|y)$ as opposed to the marginal distribution $p(x)$. For this and many other reasons, the GAN is flexible to the input and output data as long as it is treated as an image-to-image translation problem and the data distributions are similar. This has caused the architecture to be used both for classification and style transfer. More recently, the GAN has been used for image-to-image translation of aerial imagery. Therefore, this work seeks to investigate the use of cGANs for urban landcover prediction with a custom dataset.

Data Selection

High-resolution aerial imagery is difficult to access without special privileges or by paying for services. In recent years, many companies and nations have increased the resolution and frequency of satellite imagery. Some commercial imaging satellites provide high-resolution imagery, such as Worldview-3 by Digital Globe⁷. However, the cost of this data makes it infeasible for this research. For example, if the entire region of Washington DC was required, then the total cost for the 179 km² would be approximately \$2,506. At the estimated rate of \$14 per km², the entire United States would be over \$100 million. More information on

⁷<http://worldview3.digitalglobe.com/>

Table 3.6: Aerial Dataset Options

	Data	Resolution	Accessibility	Scale
Geo Data Hubs	Mostly visual spectrum	From 3 inches to 1 meter	Dependent on providers (ex. DC collected every 6-7 years)	Limited locations, ex. DC, Vermont
Copernicus Sentinel-2	13 spectral bands: visible and NIR at 10 meters, red edge and SWIR at 20 meters, and atmospheric bands at 60 meters resolution		Global 5-day revisit frequency	
LANDSAT-8	Eleven bands	30-meter	Global two-week revisit frequency	
NAIP	Red, Green, Blue, Infrared	1-meter	Three-year cycle during agricultural growing season	Continental United States

the cost of satellite imagery is available from LandInfo⁸. An overview of the aerial dataset options that are available at no cost is provided in Table 3.6. The NAIP imagery source is used for this research since it provides a limited set of inputs and an above-average resolution, at 1-meter, that should be available from other sources for humanitarian or national security applications.

3.5.2 Approach to Urban Labeling

The assumption has been that open-source data is used, primarily from Geo Data hubs. Since this data was generated in the past, it is not 100% accurate and will not be available over the entire region of interest for every class. In other words, the labeled data for supervised learning that is available has limitations in quantity, quality, and resolution.

A deep learning approach is selected because of deep neural networks' ability to understand and classify visual and multispectral data. Networks can perform efficiently even when there are many features and large amounts of data. Since the dataset is limited and may contain classifications that have not been validated, the choice is made to use a Conditional GAN, or Generative Adversarial Network. The model architecture is selected to be the Pix2Pix [88] conditional GAN with the proven U-Net encoder-decoder for semantic segmentation. Semantic segmentation allows classifications for each pixel and is, therefore, able to localize and distinguish borders. The Pix2Pix process, shown in Figure 3.20, requires a generator and a discriminator network, or an encoder-decoder network. The Pix2Pix model in this work made use of a modified U-Net [87] for the generator, and the

⁸<https://landinfo.com/>

original PatchGAN classifier detailed in the Pix2Pix paper [88].

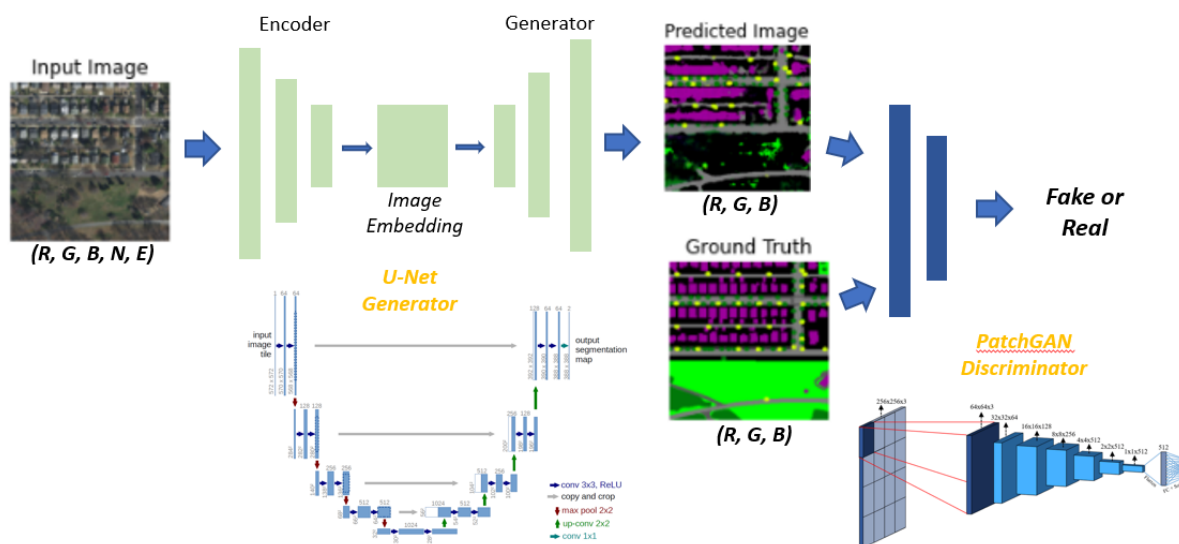


Figure 3.20: Pix2Pix Model

Process and Tools

The resolution of the predictions and the large-scale areas of interest results in computationally expensive datasets. Therefore, the Google Cloud ecosystem is selected to host the majority of the data management responsibilities. This also demonstrates that anybody is capable of performing similar analyses with a simple laptop and internet connection. The Google infrastructure provides many tools that are useful for the following research. The Google infrastructure is utilized primarily because of the access to geospatial data analysis with Earth Engine, GPUs through Google Colab, and data storage through Google Drive. The full suite of tools is seen in Figure 3.21 which shows the flow of data through the software. TensorFlow is utilized for machine learning and neural network training. It is a Python library set up well to work with the Google pipeline of data and training. TensorFlow leverages automatic differentiation using symbolic derivatives to solve the backpropagation updates quickly, particularly on GPUs. Furthermore, it has direct access to storage tools like Google Drive and Google Buckets, removing any local data passing step.

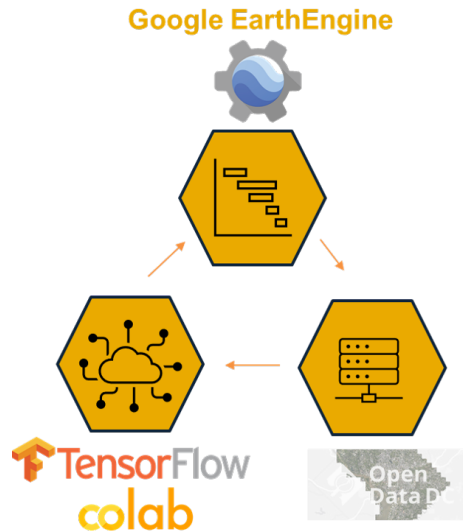


Figure 3.21: Process and Tools used for Storage, Prediction, and Visualization

Data and Model Preparation

Data is sourced from open-source Geo Datahubs like the Washington DC Open Dataset⁹. A mix of raster and vector data is transferred to the Google Earth Engine cloud, where cloud servers perform reprojection, rasterization, and resampling. The data is compiled into a training dataset using *FeatureCollection* objects in Google Earth Engine and then converted to *TFRecord* Tensorflow data structures.

The input channels for an image-to-image translation are generally the red, green, and blue, RGB, labels of a visual spectrum image. This work leverages these inputs as well as infrared and relative elevation change data. The visual spectrum channels are accessed through the National Agriculture Imagery Program, NAIP, dataset. While the data is only updated every three years, the assumption is made that similar datasets can be acquired by scouting missions or from paid-satellite datasets. Elevation data is accessed from the USGS 1-meter resolution Digital Elevation Model (DEM). This totals five input channels and three output labels that map to nine total classes, detailed in the next section. All the data is resampled to a 1-meter resolution. The data is selected from three zones of interest

⁹<https://opendata.dc.gov/>

Table 3.7: Geometric Information of Data Zones for Training and Evaluating Network

Area km^2	Zone Coordinates	Location
54.93	[-74.02338505859373, 40.77270079459629], [-74.02338505859373, 40.71339424172864], [-73.92450810546873, 40.71339424172864], [-73.92450810546873, 40.77270079459629]	New York City
23.80	[-76.99174342918374, 38.915923048582165], [-76.99174342918374, 38.9034336851992], [-76.97191654014566, 38.9034336851992], [-76.97191654014566, 38.915923048582165]	Washington DC
64.70	[-80.46629978204919, 25.905113119924145], [-80.46629978204919, 25.72833302279588], [-80.137396583807, 25.72833302279588], [-80.137396583807, 25.905113119924145]	Miami
22.05	[-84.43555103393705, 33.807222834172975], [-84.43555103393705, 33.77012946510827], [-84.37770114990384, 33.77012946510827], [-84.37770114990384, 33.807222834172975]	Atlanta

that are documented in Table 3.7.

The data labels for Washington DC are seen in Figure 3.22 and include labels for structures, greenspaces, roads, parking lots, sidewalks, water, trees, and poles. The classifications are selected based on a combination of rapid urban map need, datahub availability, and predicted influence in spatial relationships. A rasterized image of the training data is shown for verification. The urban feature labels are mapped to red, green, blue (RGB) values. The mapping to this 3-dimensional latent feature space is hypothesized to improve training in this problem since previous investigations showed promising results, as in [48]. Later, the results are compared for both RGB labels and one-hot encodings for a subset of the nine classes.

Insight can be gained for visualization of the distribution of labels and channels of the training data. The distribution of the input channels and the output classes for the Washington DC zone is shown in Figure 3.23 and Figure 3.24. Note that the elevation data is shifted far to the left since the location selected was near sea level and mostly flat. The visual spectrum channels exist mostly between 80 and 220. In Figure 3.24 the distribution of labels has been converted into the respective classes. It is clear that there

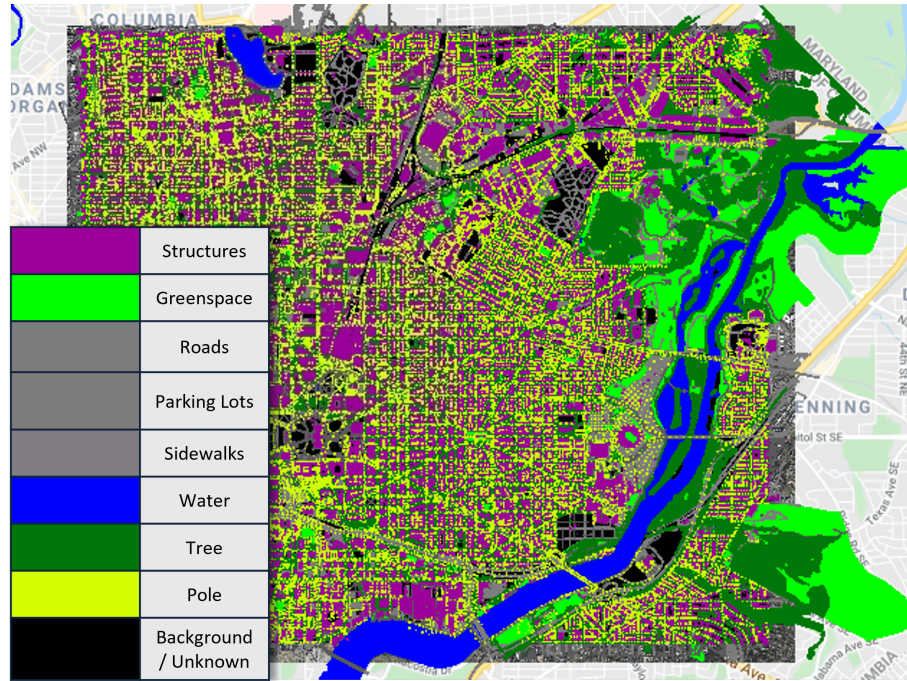


Figure 3.22: Urban Cover Labels for DC Area

are a lot of unknown/background labels. However, for individual 256×256 patches, the unknown pixels are distributed throughout, as is shown in later examples. Furthermore, some patches will be filtered if the number of unknown pixels is high. The channel and class histograms for NYC are found in Figure 3.25, Figure 3.26 and for Miami are found in Figure 3.27, Figure 3.28. Through the combined datasets, there is good coverage for both the input channels and the output labels. It is expected that a large portion of the United States would fall into a distribution similar to these three datasets.

The data is divided into subsets to keep data for tracking the training progress and evaluating the generalized performance. 90% of data is used, with the 10% left out for further verification in later chapters. Of the remaining data, 80% is used for training, 10% for evaluation, and 10% for testing performance. That data is divided into smaller samples and optimized for training with TensorFlow using TFRecord files. Data is randomly sampled from the three dataset zones from Table 3.7. The data is accessed from a Google EarthEngine ImageCollection object and converted into the target, label TFRecord files for

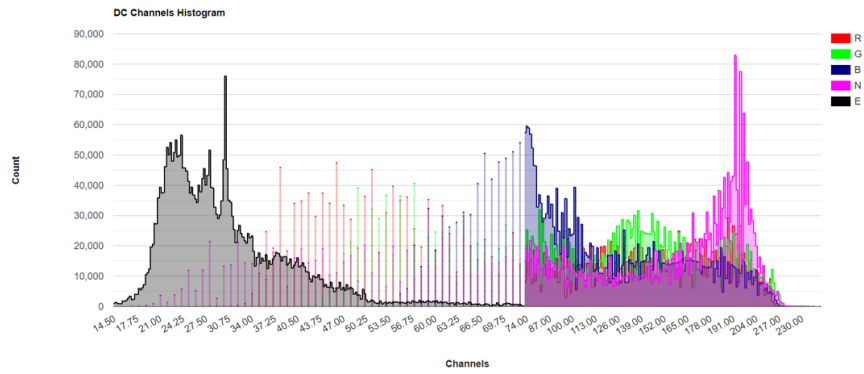


Figure 3.23: DC Channels

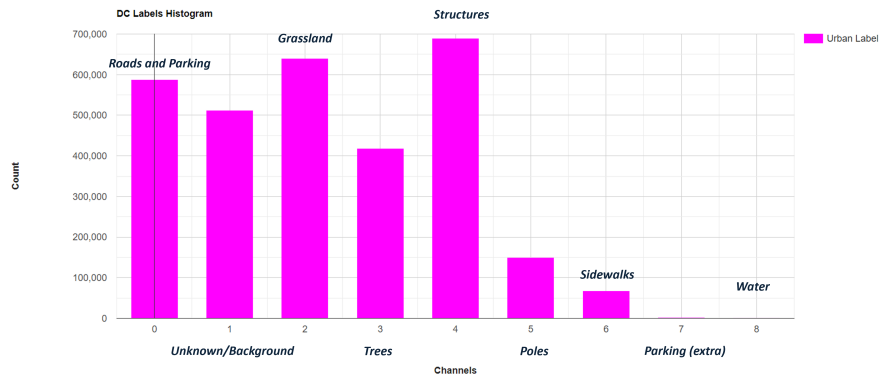


Figure 3.24: DC Classes

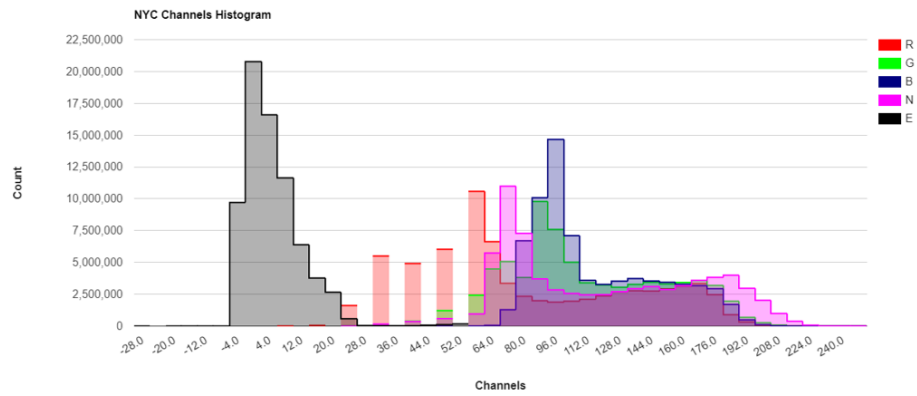


Figure 3.25: NYC Channels

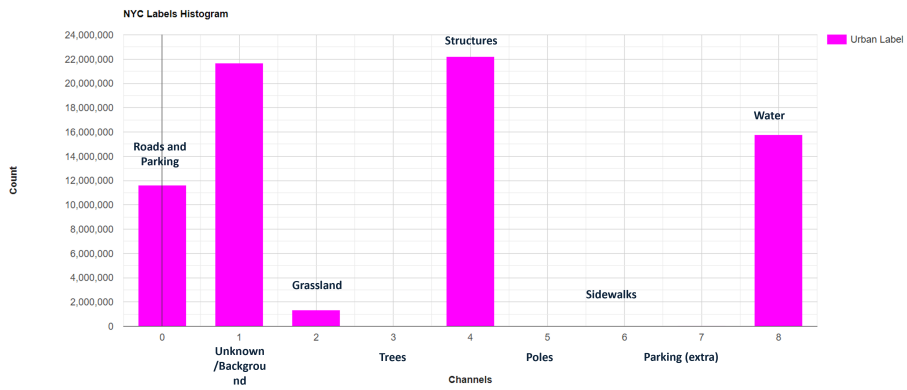


Figure 3.26: NYC Classes

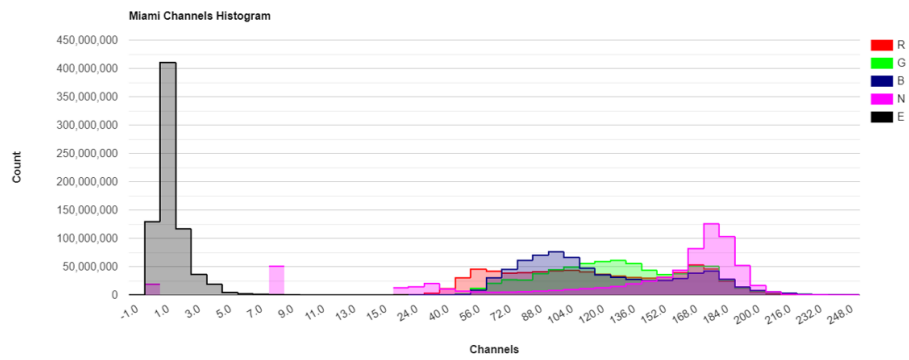


Figure 3.27: Miami Channels

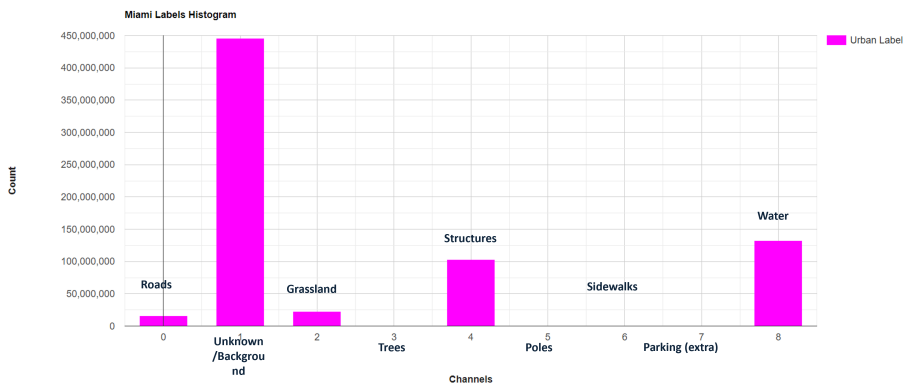


Figure 3.28: Miami Classes

training. First, the data is converted into height, width, and channel (H,W,C) dimensions, then the image is resized to 286 x 286 and randomly cropped back to 256 x 256. This adds random jitter to the dataset as documented in similar tasks. The image is normalized, with the channels (R,G,B,N) converted from bytes in (0,255) to (-1,1). The elevation channel is normalized in each batch using the minimum elevation, e_{min} , value, with the range converted from $(e_{min}, e_{min}+100)$ to (-1, 1). The class labels are normalized to the range (-1, 1). Next, 50% of the data and labels are mirrored for data augmentation. Lastly, poor samples are filtered out for the two cases where a majority of the 256 x 256 image is background or water. A ratio of 40% is used for the filtering decision.

The Pix2Pix architecture is extensively detailed in the original paper [88]. The generator is the U-Net architecture originally developed in [87] with slight modifications. The U-Net model is built from a downsampling encoder and an upsampling decoder, thus giving it the "U" name. The discriminator is the PatchGAN classifier, similar to the one from [106]. The PatchGAN discriminator attempts to classify if a 30 x 30 patch of an image sample is real or not real. It attempts this task on both a real, or target, image and a fake, or generated, image. This promotes the competitive nature of the generator to improve until the discriminator cannot tell the difference between a real or generated label. The generator and discriminator models, as produced by TensorFlow, are seen in Figure B.1.

Training the Pix2Pix

The training process involves 256x256 pixel patches at a 1-meter resolution, meaning each patch is 256 squared meters in area. The CNN kernel remains at a 256 size as well, and the dataset is compiled with a batch size of 1, as recommended in the original Pix2Pix paper. The discriminator overwhelms the generator in a few training epochs if a larger batch size is used. The loss function from the Pix2Pix paper is used, which includes both generator and discriminator loss functions. The generator loss function is a combination of a min-max optimization function seeking to minimize the generator loss and maximize

the discriminator loss, seen in Equation 3.14. A sigmoid cross-entropy loss or binary cross entropy is used to learn the weights, and is seen in Equation 3.13. The multi-label cross-entropy for label x is summed over all labels for binary cross entropy.

$$B(x)_{multi} = \sum_x \{-(p(x) * \log(q(x)) + (1 - p(x)) * \log(1 - q(x)))\} \quad (3.13)$$

$$G^* = \arg \min_G \max_D \mathcal{L}_{cGAN}(G, D) + \lambda \mathcal{L}_{L_1}(G) \quad (3.14)$$

The discriminator loss function also leverages a binary cross entropy over the output. This includes both a real loss and generated loss, from the real images and the generated images respectively. For both the generator and discriminator, the Adam optimizer is used because of past proof of the success of the algorithm. Adam is a first-order gradient-based optimization algorithm that improves stability and convergence through adaptive estimates of lower-order moments.

Google Colab provides GPUs that are highly capable of learning tasks. The Tesla P100, with a CUDA GPU Compute¹⁰ capability of 6.0 with 16 GB of RAM is used for training. The cloud architecture is leveraged for efficient training and uses consistent storing of checkpoints of the architecture weights. The discriminator loss, generator loss, and sub-components of GAN and L1 loss are computed over the evaluation dataset and tracked over training time to check performance and stability. The goal when training GANs is to balance the Discriminator and Generator training loss. In other words, neither should be winning the game. The target should generally be 0.69 for each, representing a $\log(2)$ likelihood or perplexity of 2. This means that the discriminator is equally uncertain whether the image is fake or real.

The results from the trained model are seen in Figure 3.29 and confirm the success in

¹⁰developer.nvidia.com/cuda-gpus

learning the output distributions that match the ground truth labels. For one, the distinction between the background and structures, as well as between greenery and water, is successful. On the one hand, the training for this network failed to find the pole locations, as shown in the second row, second column. However, correct labels of the core classes like water are predicted even when the training dataset is lacking from the water predictions. Therefore, the GAN is performing as expected and forming a good output that matches the distribution of the input data even under noisy labels.

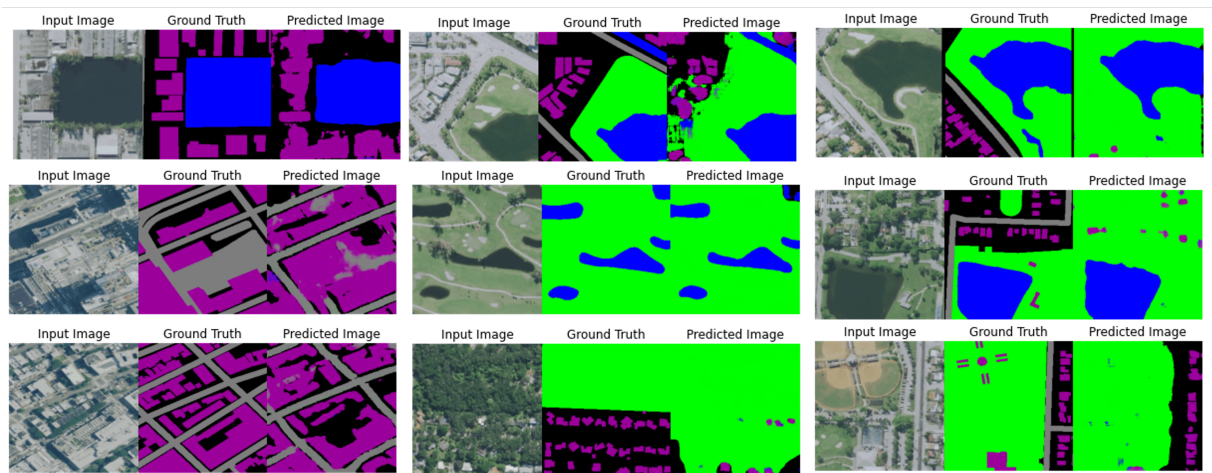


Figure 3.29: Results for Pix2Pix Model on Prediction Data using RGB Encoding

Care must be taken when training the Pix2Pix on a noisy dataset. For example, if training is allowed to continue for too long, then the discriminator will become an expert at predicting whether an image is real or fake. An overtrained discriminator limits the GANs capability and can cause instability in training. Therefore, improvements in the dataset or additional tricks during training should be investigated for improvements, and this is discussed further in the conclusion.

Evaluating the Pix2Pix

After the Pix2Pix has been trained, the three-channel encoding of the classes in the RGB structure must be mapped to the class labels. This provides some flexibility in class predic-

Table 3.8: Class Levels

	Label Encodings	Label Names
Level 1	[[0,255,0], [0,0,255],[128,128,128],[90,90,90], [125,125,125],[155,0,155],[0,120,10],[205,255,0], [0,0,0]]	[['parks','water','parking','sidewalk', 'roads','buildings','trees','poles', 'unknown']
Level 2	[[0,255,0], [0,0,255],[125,125,125],[155,0,155], [0,0,0]]	['greenery','water','concrete','buildings', 'unknown']

tion since the outputs are in a higher-dimensional feature embedding space. A visualization of the idea is shown in Figure 3.30. The labels can be selected by using a distance-based metric. For example, consider the case of a single sample compared to three classes, the purple, red, and blue crosses. Take the L2 distances to each class to be $[1, 8, 7]$. Three techniques could be used for the evaluation of the encoding to class. First, the closest label could be selected, resulting in a one-hot encoding of $[1, 0, 0]$. Second, a distance-based likelihood metric could be used such as $\frac{1/d_i}{1/\sum_j d_j}$, which results in likelihoods of $[0.79, 0.11, 0.9]$. Lastly, the softmax function could be used to apply a normalized exponential function and produce likelihood metrics for each class through the function $\sigma(z_i) = \frac{e^{z_i}}{\sum_j e^{z_j}}$ where $z_i = 1/x_i$. This results in likelihoods of $[0.54, 0.22, 0.23]$. The following results use the distance-based likelihood metric since it is the simplest to visualize. The mapping from encoding space to the 1-dimensional class space is shown with the two class levels defined for experiments, seen in Table 3.8.

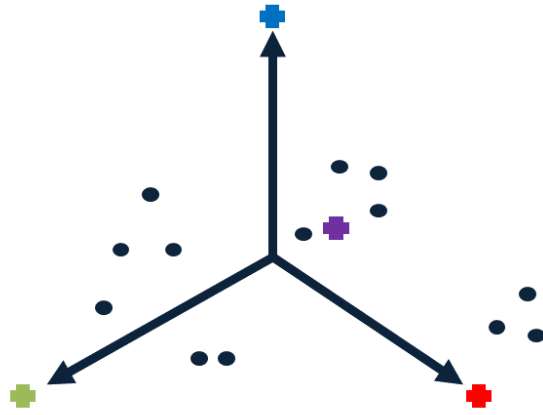


Figure 3.30: Mapping RGB Embedded Labels to Classes

Four quantitative evaluation metrics are used, seen in Table 3.9. Each evaluation metric

Table 3.9: Evaluation Metrics for Semantic Segmentation Predictions

Evaluation Metric	Evaluation Calculation
Pixel mean-square error	$(1/n)(1/m) \sum_{i=1}^n \sum_{j=1}^m (Y(i, j) - \hat{Y}(i, j))^2$
Pixel Accuracy	$TP_i + TN_i / (TP_i + TN_i + FP_i + FN_i)$
F-score	$2 \times \text{precision} \times \text{recall} / (\text{precision} + \text{recall})$
Mean Intersection-Over-Union	$TP_i / (TP_i + FP_i + FN_i)$

Table 3.10: Prediction Set Performance Metrics for Level-1 Classes with RGB Labels

	Parks	Water	Parking	Sidewalk	Road	Bulding	Tree	Pole	Unknown
Per Class Pixel Accuracy	89.69	83.80	0.00	0.50	55.40	83.89	0.27	0.01	55.88
F1 Score	0.86	0.85	0.00	0.00	0.65	0.80	0.00	0.00	0.58
IoU Score	0.91	0.98	0.97	0.99	0.90	0.84	0.99	0.99	0.78

has pros and cons in its use. For example, The F-score is not good at comparing between different methods. At the same time, the mean squared error is not informative about the ability to generalize to other datasets, as explained in [88]. All three metrics are used in the results, and successful models are expected to perform well across all three metrics.

3.5.3 Network Performance

The results after training for over 200,000 epochs are seen in Figure 3.29. The model can successfully predict buildings and parks, the core classes for urban mapping. The performance metrics for pixel accuracy, F1 and IOU scores are in Table 3.10. The IOU score and F1 score hover around 80%, showing a quality output over a dataset with a similar distribution to the training data.

Performance in the smaller classes, such as tree or pole, is unsuccessful. This is expected since the majority of these classes were only available in the Washington DC dataset. The RGB encoding allows the number of classes to be reduced by simply remapping the

Table 3.11: Pix2Pix RGB to 5 Labels Confusion Matrix Values

Numbers x 10 ⁶	Greenery	Water	Concrete	Structures	Unknown
True Positives	19.0	27.0	23.7	18.0	19.7
False Positives	1.78	0.21	0.88	1.70	2.09
False Negatives	0.39	0.18	1.18	1.70	3.26
True Negative	7.32	1.13	2.8	7.10	3.50

Table 3.12: Evaluation Metrics for RGB Output Pix2Pix Model for Level 2 Classes (5 Labels)

	Greenery	Water	Concrete	Structures	Unknown
Per Class Pixel Accuracy	94.95	86.39	70.14	81.13	51.73
F1 Score	0.87	0.85	0.73	0.81	0.57
IoU Score	0.90	0.99	0.92	0.84	0.79

Pix2Pix outputs to new a new set of labels. For example, the labels can be reduced to the five key classes where the network has shown success. The classes are selected for parks, water, roads, buildings, and unknown, and classified as greenery, water, concrete, structures, and unknown. The results after training for the same number of epochs as the previous model are seen in Figure B.2, and the evaluation metrics are detailed in Table 3.12.

The results improve the performance of greenery, water, and concrete, while resulting in small changes in structures and unknown labels. Since these are the classes that performed well in the 9-class predictions, the results are expected. For example, the training data samples can be visualized in the RGB encoding space with the 9- and 5-class labels. In Figure 3.31, samples from the training data on the right is shown in a 3D plot and compared in L2 distance to each of the nine classes. Then, four of the five level-2 classes are shown forming groups with samples in regions of the 3D space. The results show issues with water predictions, which are actually building shadows. Some samples exist in the color space between water, concrete, and structures, which potentially could indicate a new class related to shadows. The approach could provide insight into unknown classes by using clustering algorithms to find new, untrained classes, but this is left for future work.

Comparison to One-hot Encoding

The hypothesis that the RGB encoding improves results is proven true by the experiment. The Pix2Pix model and data are reconfigured by altering the three channels for the training data to have five channels for five unique classes in a one-hot encoding. The Pix2Pix model is modified to output the likelihood of each class. The five level-2 classes are se-

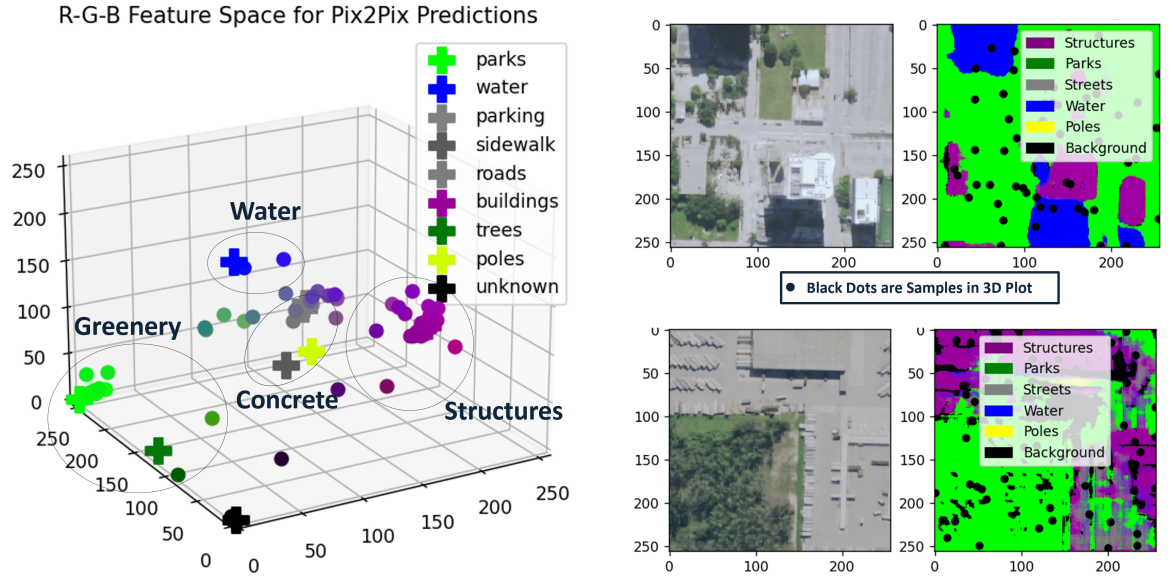


Figure 3.31: Mapping Samples of Test Data to Classes in the RGB Encoding Space

lected in place of the nine level-1 classes to reduce the computational load for training. Table 3.13 details the results, where performance is reduced as compared to the RGB encoding training. The important classes of greenery and concrete see over a 50% reduction in pixel accuracy, while structures and unknown see about a 50% reduction as well. Structure predictions from the one-hot encoding training have a decent F1 and IOU score, but the reduction in accuracy is clear from examining the visualization of the results. The outputs are less precise, with structure predictions having less detailed borders. Furthermore, the one-hot encoding causes a failure to predict the unknown class. This could be fixed by first reducing the filtering done before training, where large portions of unknown labels were removed. Alternatively, the model could ignore the unknown channel and take the softmax of the other channels' predictions. By setting a prediction threshold, any predictions with a likelihood less than the threshold on all classes would fall into the unknown class. It is expected that this would improve the unknown predictions and the other classes. However, the performance cannot improve enough to match the RGB encoding model's accuracy. More data and additional tracking tricks are expected to improve both models. The hypoth-

Table 3.13: Evaluation Metrics for 5-Channel One-hot Encoding Output Pix2Pix Model for Level 2 Classes (5 Labels)

	Greenery	Water	Concrete	Structures	Unknown
Per Class Pixel Accuracy	6.36	74.98	18.39	48.56	0.00
F1 Score	0.12	0.51	0.29	0.62	0.00
IoU Score	0.49	0.86	0.89	0.82	0.39

esis is confirmed and shows that training a GAN on noisy datasets with limited accuracy can be improved by mapping to a lower-dimensional feature space instead of training many channels in a traditional one-hot encoding method.

Comparison to Benchmarks

A comparison between the classes of NLCD and Google DW is shown in Figure B.3. This mapping can be used to directly compare labels of the truth and prediction labels to gain insight into the information from each. The comparison between the predictions is visualized in Figure 3.32 and color legend information can be found on the data sources in Google Earth Engine¹¹. The results show improvements that are evident in the resolution and detail of predictions. The Pix2Pix predictions are 30 times the resolution of NLCD and ten times the resolution of GDW. NLCD and GDW are unable to provide detailed structures, and the GDW shows incorrect greenery and water predictions in the Atlanta data. The areas where the benchmarks perform well are with precise greenery from NLCD and detailed road types from NLCD's impervious classes. In general, the large number of classes the two benchmarks are able to predict is powerful but more useful for large-scale analysis or more general landcover maps.

3.5.4 Results

The output of the Pix2Pix network is an image of labeled pixels, which must be transformed into vector objects for efficient storage and use in the RUM framework. Each pixel must be matched and shaped into polygons to speed up any spatial calculations and for faster

¹¹<https://developers.google.com/earth-engine/datasets>

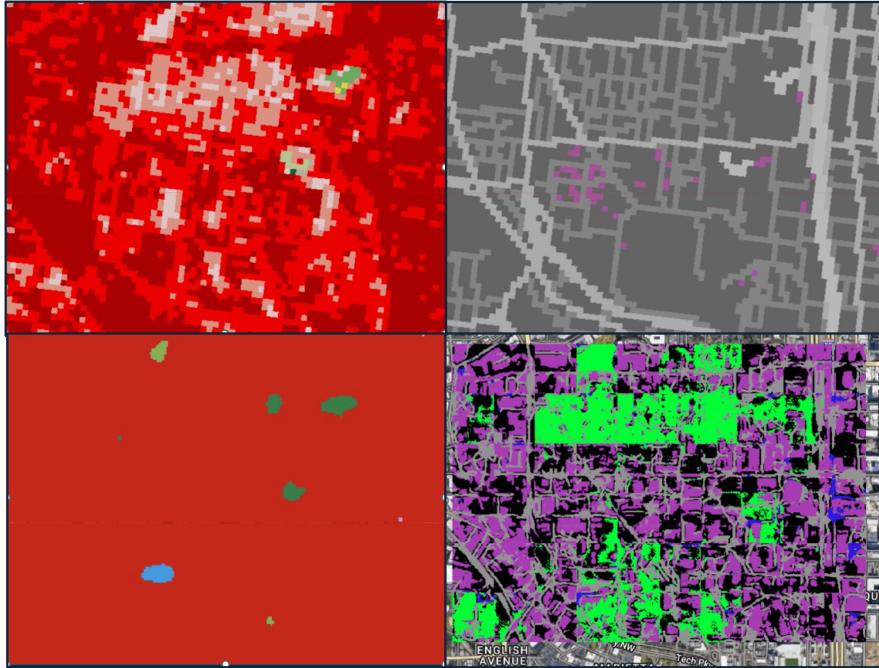


Figure 3.32: Comparison of labels for NLCD (top left), Impervious Labels for NLCD (top right), Google Dynamic World (bottom left), and Custom RUM Labels (bottom right)

visualization. Traditionally, the solution to this problem is called vectorization, or image tracing. Image tracing is often referenced for applications such as converting JPEG images to SVG graphic files. At the same time, vectorization is often the name for converting large spatial datasets from raster to vector files. The simplest method is to track continuous paths of pixels that match the same class and then fit with one or more polygons. If the pixels do not have clear boundaries, then clustering may be needed before vectorization, as demonstrated in work by the author in [75].

The total pipeline is now demonstrated for the Atlanta area of interest from Figure 3.1 and is compared to the previous RUM model in the FME workbench. The FME city map model using the Pix2Pix feature labels with the 3D buildings extruded from the structures label is seen in Figure 3.33. The height of the discretized building polygons is approximated using the nearest height value in the open-source structure dataset. The colored labels in the FME model come from the Pix2Pix predictions, which can be understood better using

a histogram of predictions in Figure 3.34.

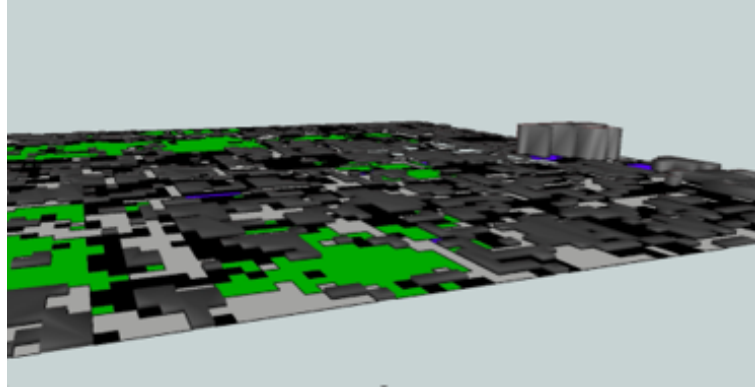


Figure 3.33: Atlanta Model for 3D Structures and Urban Feature Labels in FME Software

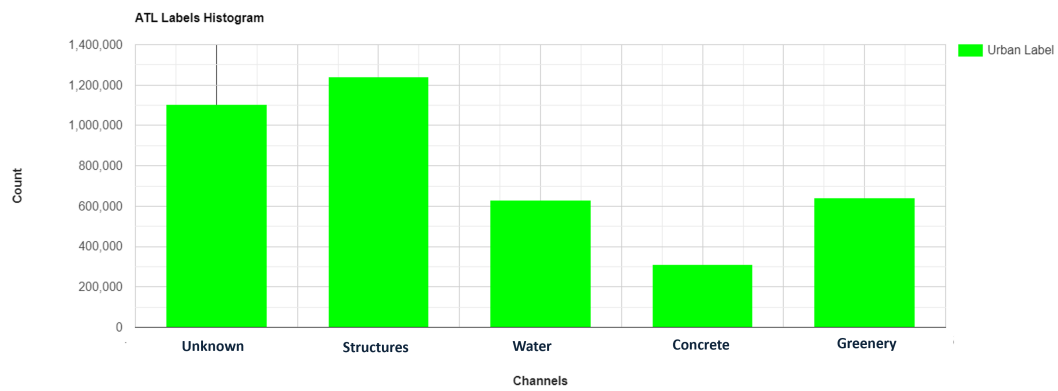


Figure 3.34: Atlanta 5 Channel Predictions Histogram

3.6 Summary and Future Work

A rapid urban map method named RUM is demonstrated to successfully represent a region of a city using open-source or predicted datasets. Open-source data is collected and transformed into an efficient storage and query source for trajectory planning in 3D. The accuracy of the maps is within the minimum bounds if a discretization is selected with a high threshold and small cell size for the structures or if buffers for the bounding box are allowed, which results in 60% more volume, but fully encapsulates structures 99% on

average. The terrain is modeled with the NURBS or Gaussian Process since they show improvements on the simpler B-spline model for mountainous terrain. GP has the smallest average error and provides the unique capability of Bayesian outputs. The building, terrain, and weather models are also constrained by parameters that are controlled by computational limitations.

If data is missing then the Pix2Pix GAN is able to predict with aerial or satellite imagery, which is converted into the same format and used in the same framework. Furthermore, additional features are available to identify key areas or risky flight paths. Structure prediction was successful with scores of about 0.80 for pixel accuracy, F1, and IOU score, while additional features such as greenspaces, water areas, and roads scored just below this level. The FME Workbench tool is used for verification of the structure models, the terrain matching, and the urban landcover prediction labels. Visual verification and validation with a cityGML model of New York City confirm success in urban map model, which is seen in Figure 3.35. The 3D urban map models are now available for subsequent experiments and the following chapters address how to plan trajectories within these urban maps.

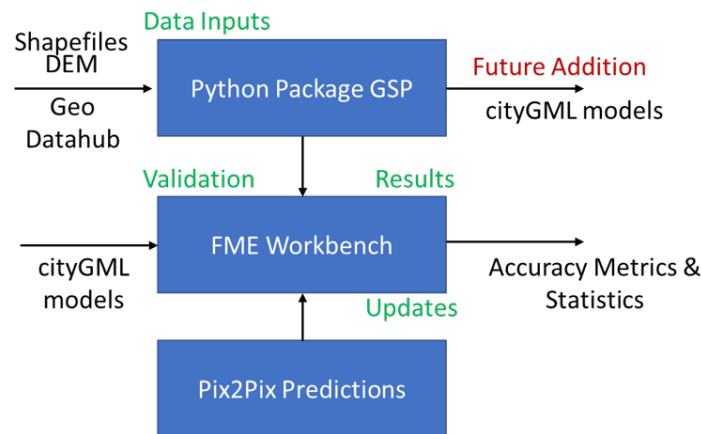


Figure 3.35: Pipeline for Evaluation of Urban Map Model using FME Benchmark

Future additions to the RUM framework include improved training for the Pix2Pix model, fusing additional data, and taking up the cityJSON standard. Pix2Pix training issues include non-convergence, where the models do not converge, mode collapse, where

the generator produces limited modes, and slow training, where the gradient to train the generator vanishes. One solution is to use instance noise, or regularization [107] to improve. Suppose the real data distribution, \mathcal{P} , and the data distribution of the generative model, \mathcal{Q} are not overlapping well. In that case, the discriminator can start to predict real vs. fake perfectly. This leads to instabilities in training as the generator cannot change weights productively to beat the discriminator. One solution to this is to add a Gaussian model of instance noise. This is expected to expand the real and generated distributions to achieve significant overlap between the two. As expected, the noise should make the discriminator's job harder and make it more difficult to predict real and fake. In [107], Roth shows that the same effect can be achieved by regularizing the norm of discriminator gradients instead of adding noise to samples directly. There are even more advancements for improving cGANs other than convergence modifications, such as adding a dual-branch structure, as explained by Liu et al. in [108]. In addition, more training in new and unique distributions of input and output data will improve generalized performance. There exists additional data for each of the training locations that could be labeled or validated, and more data is easily accessible through verified OpenStreetMap datasets or from other Geo Datahubs like Los Angeles, Denver, and Houston. In addition, future work should leverage the cityJSON and cityGML standards for benchmarking and model comparisons. Then, future validation and verification could be improved by leveraging tools for CityGML or CityJSON like va3lidity and 3d-building-metrics.

EFFICIENT THREE-DIMENSIONAL OFFLINE KINODYNAMIC PLANNING

4.1 Introduction

Aerial system trajectory planning has advanced rapidly in the past 20 years from the increased availability, applicability, and reliability. As mentioned in earlier chapters, there is an extensive list of algorithms to solve the trajectory problem. The accuracy, mission type, and objectives are factors in the choice of an algorithm. For example, the rocket landing problem, successfully demonstrated by SpaceX, has been solved mostly through convex optimization problems derived specifically for rocket dynamics during landing. The convexification of the problem, as detailed in [45], is key to the modern-day success.

In this work, the goal is to find optimal trajectories through the use of an offline planner that can provide insight into how aerial systems can respond to disaster scenarios. Disaster response missions require aerial systems to maneuver through the complex environment to collect data and provide situational awareness for first responders. Aerial systems' capability to provide situational awareness has been proven in the past by piloted demonstrations and autonomous research experiments. However, specific insights into the system capability and performance are needed to maximize mission success and first responder safety.

This chapter focuses on the kinodynamic planner in low-altitude, 3D environments. First, an argument is made for the Stable-Sparse RRT algorithm. Next, the algorithm is put to the test against the kinodynamic version of RRT with multiple generated maps and using a quadcopter dynamics model with four control inputs sampled for open-loop control. The SST algorithm is then investigated for the best performance for the maps created from Chapter 3. The algorithm is evaluated over multiple urban maps and with external wind and internal noise parameters. The computational requirements of the dynamics model are

investigated and an improved method for small-scale maps is demonstrated.

Trajectory Planning Background

Consider a system modeled with continuous dynamics that is defined by differential equations, with vector variables for the state, x , and control, u , a scalar variable for time, t , and differential equations for the state evolution $f(x, u, t)$. The state and control are bounded by the state space \mathbb{X} and control space \mathbb{U} . An accurate model of the system may require a high-dimensional representation of each space, either through a discrete or continuous domain, the latter of which is represented in Equation 4.1.

$$\dot{x}(t) = f(x(t), u(t)), \quad x(t) \in \mathbb{X}, \quad u(t) \in \mathbb{U} \quad (4.1)$$

The first target is to determine whether a feasible path or trajectory exists at all. For a feasible path to exist, there must be a set of controls u_0 to u_k or a control policy $u(t)$ that takes the system from the initial position, x_0 , to a position in the goal set, \mathcal{X}_{goal} . The set of controls that achieve this is shown in Equation 4.2.

$$\{(u_0, \dots, u_k) \in \mathcal{U} \mid x(t) = \int_0^t f(x(\tau), u(\tau)) d\tau \in \mathcal{X}_{goal}\} \quad (4.2)$$

A feasible path is defined as the set of controls and states that can be achieved from the initial state of the system to the goal state. An additional constraint is the avoidance of obstacles guaranteeing a collision-free path. The environment, \mathbb{Q} , is defined by geometric primitives or collision functions in \mathcal{R}^3 space. A map can be discretized during planning to consider each obstacle, o_i , for collision checks. In simple terms, a collision check is a function that compares the current and future states of the system and determines if the system will touch one or more obstacles.

A trajectory path that exists is defined as $\sigma(t)$. An additional parameter of a trajectory path is that the states $x(t)$ must not enter the space of obstacles, \mathbb{X}_{obs} , and the controls $u(t)$

must stay in the bounds of feasible control inputs, $\mathbb{U}_{feasible}$.

$$\sigma(t) = \int_{t=0}^{t=t_f} f(x(t), u(t)) dt$$

The motion planning problem seeks to find a set of actions, or controls, to move from one configuration to another. This depends on the model for both the system, x , the environment, \mathbb{Q} , and the controls, u . An optimal path must meet all the previous constraints of feasibility and collision-free paths while extending this to comparing all possible paths with a single cost function, $J(x(t), u(t), f(x, u, t))$. The optimal path, σ^* , is guaranteed to be better than all other paths for a given cost function, as defined in Equation 4.3.

$$\sigma^*(t) = \underset{u(t)}{argmin} \int_{t=0}^{t=t_f} J(x(t), u(t), f(x, u, t)) dt \quad (4.3)$$

The cost function or objective function, named by pessimists and optimists respectively, evaluates a trajectory through the state, control, or other temporal parameters that are stored along the trajectory. An initial approach the the trajectory problem commonly uses the $\ell^2 - norm$ or L^2 vector norm. Also used for euclidean distance calculations, the L^2 norm is calculated over the length of the vector by Equation 4.4, where x is the vector connecting two points x_1 and x_2 and $d(x_1, x_2)$ is the euclidean distance formula.

$$\|x\| = \sqrt{\sum_{k=1}^n |x_k|^2} \quad \propto \quad d(x_1, x_2) = \sqrt{\sum_{k=1}^n (x_1^i - x_2^i)^2} \quad (4.4)$$

More advanced cost functions measure differences directly in the control variables or other metrics related to energy, time, or mission performance. Aerial system trajectory planning most often looks to optimize on one or more of these metrics, as will be introduced later in the chapter.

Aerial Flight Planning

A flight trajectory generated for disaster response must meet specific standards, as discussed in the previous section. These can be thought of as constraints in the optimization problem. For one, the flight must be feasible for the dynamic system to achieve. For example, consider the energy of the aerial system during flight. The available expendable energy from a Lithium-Polymer battery on standard unmanned aerial systems is limited, and the exact value may not be known. Next, the path should be optimal upon arrival at the goal location. The goal may be defined as a region with a bound around it to ensure success even when an exact state cannot be achieved with an acceptable error.

Optimality is a subjective factor that may change based on the mission or stakeholder. Examples include time, distance, energy, and safety. These metrics must be easily calculated, given the trajectory search algorithm. In addition, the solution of one or more paths must meet the computational requirements. This may lead to limits on the computational time, which would prioritize the finite-time performance of the algorithm.

Approaches that simplify the dynamics and solve the constrained optimization problem have been implemented in the past, such as work by Cinar in [109]. In addition, alternative methods that look to simplify the approximate the system along a finite horizon problem at discrete time steps is seen by Lee in [110]. The issue remains that the high-dimensionality of the state and control space of the vehicle, the nonlinearities in constraints and objective functions, and the coupled differential equations make the optimization problem complex. The problem cannot be guaranteed to have a solution without applying linearization and convexification to reduce the problem to a convex optimization problem. Therefore, alternative solutions are preferred, as further explained in the aerial trajectory planning literature.

There are many algorithms to solve trajectory or motion planning problems, such as optimization-based methods, grid-based methods, and sampling-based methods. For example, previous work in [75] and [49] examined different planning algorithms in an urban

environment at a predefined altitude. This work features high-dimensional dynamic systems that must plan trajectories in large, 3D, crowded maps with uncertainty included. Four key criteria are used to select the best set of algorithms to use for planning.

4.1.1 Algorithm Selection

Four criteria of interest are investigated to determine the best approach to the problem. For one, the algorithm must be capable of both soft and hard constraints. For example, the avoidance of obstacles must be a hard constraint to prevent any collisions, but the constraint on the distance to the goal target for a path should be soft, therefore less restrictive. Two, the approach must handle high-dimensional, nonlinear black-box dynamic functions and environmental uncertainty. Many algorithms require approximations of the dynamics and uncertainty to be able to for probabilistic solutions. Third, the algorithm needs to be successful in finite-time performance, where paths are smooth and are able to be reused or improved in the future. This work is not focused on real-time planning, but the speed of the offline planner is critical for Monte Carlo simulation speed. Lastly, access to optimal solutions, albeit in infinite time, is necessary for confidence in the planner and use under challenging problems. Sampling-based planners have proven successful in the past in these areas as discussed in Chapter 2 and are selected for investigation.

4.2 Background

Sampling-based methods leverage a stochastic technique and can guarantee completion and optimality with particular algorithms such as probabilistic roadmaps or Rapidly exploring Random Tree (RRT) [111]. Completion and optimality are important for a planning algorithm as they ensure continued improvement over wall-clock time and guarantee an eventual solution if one exists. Probabilistic completeness guarantees that over infinite iterations, the planner will find a solution if one exists. If all possible paths are Σ , the probability that a feasible path, σ , exists for the state space, start, and goal at iteration, n , is

equal to 1, as seen in Equation 4.5.

$$\lim_{n \rightarrow \infty} \mathbb{P}(\sigma \in \Sigma(\mathbb{X}, x_0, x_g, \mathbb{U}, n)) = (1 \mid \Sigma(\mathbb{X}, x_0, x_g, \mathbb{U}) \neq \emptyset) \quad (4.5)$$

Asymptotic optimality goes further and says that over infinite iterations, the best path will be optimal. The cost function is evaluated over a trajectory path, such that it takes in the set of states and outputs a scalar quantity that can be used as a measure to minimize or maximize as seen in Equation 4.6. Therefore, over an infinite set of trajectories found over infinite iterations, the best path at iteration n , σ_n , will be the best optimal path, σ^* , as shown in Equation 4.8.

$$J(\sigma^n(t)) = \int_{t=0}^{t_f} c(x(t), u(t), t) dt \quad (4.6)$$

$$J(\sigma^*) \leq J(\sigma^n) \quad \forall \quad n \quad (4.7)$$

$$P(\lim_{n \rightarrow \infty} \sigma^n == \sigma^*) = 1 \quad (4.8)$$

Some algorithms meet the standard of asymptotic near optimality, which is a less restrictive property saying that the optimal trajectory, σ^* , will be within some bound, Δ_{opt} , to the best trajectory found over infinite iterations, σ^n s.t. $n \rightarrow \infty$. Finite-time performance may be the same for an asymptotic optimal and asymptotic near-optimal algorithm, however, and the finite-time performance should be evaluated relative to the near-optimal bound, Δ_{opt} , as shown in Equation 4.9.

$$P(\lim_{n \rightarrow \infty} ||\sigma^n - \sigma^*|| \leq \Delta) = 1 \quad (4.9)$$

Sampling-based Planning Overview

The general framework of a sampling-based planner follows six steps, as detailed in [39]. First, the graph, $G(V, E)$ is initialized with the first node, V_1 , and no edges, $E = \emptyset$. The graph must ensure nodes are in C_{free} , the configuration space defined for the system where it is free of obstacles and restrictions. Second, a vertex, $v_i \in V$ is selected for expansion. This could be a random selection or biased by some measure, but a node must always have only a single parent to keep the graph as an undirected graph or tree. Third, the expansion from the current node v_i to a new node v_j forms the start, or continuation, of a path σ^n in which all nodes are in free space, see Equation 4.10. Fourth, a new edge is generated if it holds true that the path is feasible, as in Equation 4.11. Fifth, the current path, σ^n , is evaluated to see if it is a complete path from the start node, v_1 to the goal node, v_g . Sixth, the previous steps from step 2 are repeated and iterated until one solution is found, or otherwise, a termination condition is satisfied.

$$\{\sigma^n(v_1, \dots, v_g) \mid (v_i) \in C_{free} \ \forall \ i\} \quad (4.10)$$

$$e_{i,j} = (v_i, v_j) \mid feasible(v_i, v_j) == True \quad (4.11)$$

Each node, v_i , stores state information for the trajectory. The stored information may be as simple as the 2D or 3D position but could also include control and second-order dynamic states. The feasibility of paths can be checked at each expansion of the node v_i to v_j or may occur over multiple segments of a path. Each algorithm addressed each of the steps in a different way that may provide asymptotic performance guarantees or success in specific problems. The Rapidly Exploring Random Trees algorithm, or RRT, has demonstrated both theoretical guarantees and applicable success over many problems and has become a default starting point for sampling-based planners. The RRT algorithm is detailed by Lavelle in [39].

4.2.1 Rapidly Exploring Random Trees

If an exhaustive search for paths in 3D space is planned, the best 3-dimensional polynomial-time approximate algorithms have a time complexity of $2^{2^{O(N)}}$ [112]. However, RRT as detailed in [111] has log-linear time complexity $O(n \log n)$. The RRT* algorithm, seen in algorithm 1, furthered the theory by guaranteeing asymptotic optimality in [47]. The RRT* algorithm provides probabilistic completeness and asymptotically optimal guarantees. Asymptotic optimality is guaranteed according to Theorem 38 in [47] such that the ratio of the volume of the space to the volume of the unit sphere, $(\mu(X_{Free})/\zeta_d)$, is constrained by Equation 4.12.

$$\gamma_{RRT^*} > (2(1 + 1/d))^{1/d} (\mu(X_{Free})/\zeta_d)^{1/d} \quad (4.12)$$

Algorithm 1: RRT*(X, x_0, N, γ)

Data: $x_0 \in X, N > 0$

Result: $p^* = t_1, x_1 | \dots | t_f, x_f \in \mathcal{P}$

$$cost(p^*) \leq cost(p_i) \forall p_i \in P;$$
$$V \leftarrow \{x_0\} ;$$
$$E = \emptyset ;$$
$$\mathbf{G} = \{V, E\} ;$$
for N **do**
$$x_{rand} \leftarrow SampleFree(X);$$
$$x_{nearest} \leftarrow \text{Nearest}(G, x_{rand}) ;$$

```
/* Node Selection */
```

$$x_{new} \leftarrow Steer(x_{nearest}, x_{rand}) ;$$
if $CollisionFree(x_{selected}, x_{new})$ **then**
$$x_{near} \leftarrow Near(x_{new}, \gamma_{RRT*}, G) ;$$

```

 $x_{parent} \leftarrow ChooseParent(x_{near}, x_{nearest}, x_{new});$  /* Connect along
min-cost */

```

$$G \leftarrow Insert(x_{parent}, x_{new}, G) ;$$
$$G \leftarrow \text{Rewire}(G, x_{\text{near}}, x_{\text{parent}}, x_{\text{new}}); \quad /* \text{Rewire the Tree} */$$

end

end

Recent literature has revisited the RRT and RRT* algorithm to update the theoretical guarantees [113] and general performance metrics [114]. However, most of the recent lit-

erature has sought to add new features and tune the internal functions for specific problems of interest. A summary of algorithm advancements as of 2016 is found in [115]. Joshi in [116] summarized the more recent focus for sampling-based planner improvements as algorithms like RRT# [117], FMT* [118] and BIT* [119] that leverage heuristics and dynamic programming for faster convergence. In addition, algorithms like RABIT* [120] and TIE [116] improve the sampling methods through rejection sampling and reachability sets, respectively.

In addition, there has been plenty of research into path refinements, such as improving SO3 or SO2 paths using spline curves. Optimization-based refinement like CHOMP [121], GuSTO [122], and ccSQP [123] have proven successful at smoothing and improving paths. Other techniques exist that involve the controller or onboard flight, like Model Predictive RRT[124]. Moreover, the AI and neural network revolution of the 2010s has led to many applications for autonomous systems and robotics. For example, Lichter in [125] constructed three networks, including an autoencoding network, a dynamics network, and a collision checking network. These networks sought to mimic the primary functionalities of a sampling-based planning algorithm explained by Lichter to be state sampling, local steering, and collision checking.

4.2.2 Kinodynamic Planning

Vehicle Kinodynamics, as detailed in [111], are constraints for obstacle avoidance and require graph search constraints of reachability or feasibility of nodes. Recent literature exists for kinodynamic-constrained, sampling-based planning applied to UAVs in urban environments [126]. However, works such as this often leverage methods that do not provide safety guarantees or statistical confidence, like the artificial potential field and point-mass dynamic models. Some techniques include kinodynamic RRT or RRT with differential constraints, which require the feasibility of the edge transitions between nodes in the search tree. The internal procedure is called a steering function and is a topic of research for

sampling-based planning. The dynamic function can be solved using the constrained TP-BVP, which is computationally challenging. However, alternative approaches have leveraged the forward-propagation of the dynamics model directly in the planning algorithm.

The assumption is made that expensive dynamic models are necessary for the prediction of energy and maneuvering in windy conditions and that aerial system configurations vary widely. In addition, it is assumed that decision-makers and stakeholders interested in the disaster response framework may not know the configuration of the aerial systems before investigation. Therefore, forward-propagation dynamics models or black-box dynamic functions are used, and a quadcopter model is developed as a representative model.

Aerial systems that operate under kinodynamic constraints must be modeled mathematically to search for trajectories with a sampling-based kinodynamic planner. The simplest form of a dynamic aerial model could be used, assuming a point-mass in 3D with linearized dynamics about a forward flight state. However, this would limit the trajectories that could be found and would result in inaccuracies when the actual system attempts to follow the path. Traditionally, an aerial system is defined by 12 degrees of freedom through the kinematic and dynamic equations. A detailed explanation by Etkin can be found in [127]. However, the standard 12-dof vehicle with aerodynamic force and moment calculations are often computationally expensive to use, and high fidelity models are expensive to model.

4.3 Research Question 2

For aerial system disaster response, there is a need for feasible, efficient flight paths generated within the urban maps model created in Chapter 3. Sampling-based planning algorithms have proven successful in finding optimal or near-optimal trajectories with complex dynamic models. However, there has not been a systematic investigation using a modeling and simulation environment to determine the best methods for predicting flight paths in realistic environments. Therefore the primary question for this chapter is as follows.

Research Question 2

What trajectory planning algorithm consistently and efficiently finds offline kinodynamic paths in 3D while operating with unknown dynamics in uncertain environments?

The hypothesis is that sampling-based planning methods, proven in the past for field robotics, perform well under finite-time constraints and are able to generate near-optimal trajectories for both small-scale and large-scale flight missions. Sampling-based methods can find fast, efficient routes through the map using black-box dynamic models and both soft and hard constraints. Parameter optimization and algorithm improvements can lead to improved performance, and algorithm performance is limited by the availability of time and data. Furthermore, the modeling and simulation framework can be used as a benchmark demonstration for preflight collision-free, confidently safe paths for deployment in disaster response missions by demonstrating repeated flights in maps using Monte Carlo simulation.

Two experiments are set up to evaluate the hypotheses. First, a sampling-based planning algorithm is selected as a baseline method and then improved upon using a systematic evaluation of techniques to enhance individual functions. The algorithm is demonstrated in multiple urban maps and evaluated on the ability to find feasible and optimal paths in finite time. The computational load in both memory and time is compared against the baseline methods and investigated for improvements. Second, the planner's performance is explicitly evaluated with regard to the dynamic models and, thus, the kinodynamic constraints. The dynamic model's impact on the final path and the computational time to compute optimal paths is evaluated. Alternative methods for computing dynamic forward propagation through motion primitives are investigated to assess if they increase or decrease accuracy, computational load, and efficiency.

An overview of the pipeline to experiment with and evaluate the planning algorithms is shown in Figure 4.1. The pipeline takes as inputs the map, the algorithm, and the dynamics model and produces trajectory details, computational requirements, and flight performance. Stochastic results are inherent to the planning algorithm but also come from the wind and

noise models and any randomness in the map creation.

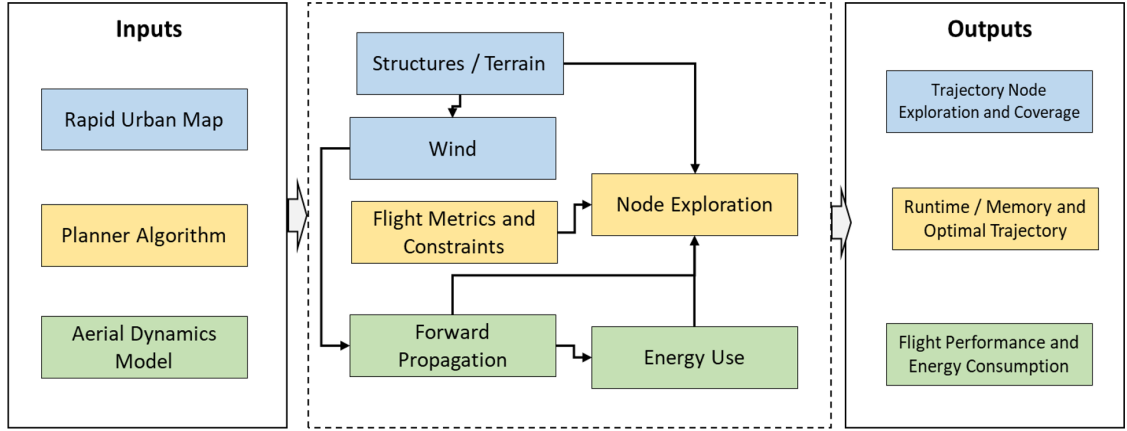


Figure 4.1: Planning Experiment Overview

4.4 Efficient and Effective Sampling-based Planner

The RRT algorithm has been widely used for aerial vehicle navigation in the past, usually with modifications to account for the kinodynamic constraints through a steering function or a forward propagation expansion. However, many methods run into convergence or optimality concerns when applied to high-dimensional models and large search spaces. As concisely explained by Choudhury in [64], the goal of the trajectory planner is to find "informative sparse likely paths" for these aerial systems. Informative, meaning paths that are likely to be optimal, and sparse, meaning spread out within the search space to limit memory and exhaust the general search space in less time. While there have been many methods to improve upon the RRT algorithm, the Stable-Sparse-RRT, SST, algorithm provides key features that are proven to work well and meet the limits and objectives of this research application. SST is well-suited for planning using kinodynamic constraints because of the Monte Carlo propagation function. The algorithm handles uncertainty in challenging environments well by finding sparse paths quickly that can be compared with statistical

confidence measures. SST has been applied for field robots with finite-time performance that still finds near-optimal paths, as demonstrated by Littlefield in [128].

4.4.1 Stable-Sparse-RRT

The SST algorithm, introduced by Littlefield in [52], can guarantee asymptotic optimality even when only a forward propagation model of the dynamics is available, thereby removing the need for an approximation from a steering function or a complex TPBVP solver. The algorithms consider the dynamics in the cost of building the tree, then forward propagate the dynamics to sample a new feasible point. The propagation methods use either the controls of the vehicle or a higher level trajectory state that uses trimmed flight conditions. These are often referred to as maneuver automata or motion primitives.

The propagation-based SST algorithm is demonstrated for vertical takeoff and landing, VTOL, aircraft in the work by Nurimbetov [129]. The vehicle plans a flight path by taking off vertically, transitioning to a fixed-wing flight model, and then planning a series of maneuvers to the goal location. The success of SST was previously demonstrated for VTOL trajectory generation in [130, 131] with example trajectories shown in Figure 4.2. The primary contribution from the work in [130, 131] was the addition of bias terms in the cost function and sampling function to produce smooth flight paths with predefined motion primitives of trimmed flight states. A 12-DOF electric VTOL aircraft was developed for forward propagation of the dynamics using cruise, climb, transition, and more flight states. Takeoff and landing flight maneuvers were added to the start and end of the trajectory using logic constraints within the algorithm.

The SST algorithm in algorithm 2 starts by initializing the priority queue sets and nearest neighbor data structures and then loops for N iterations through the functions of selection, propagation, and pruning. The *BestFirstSelection* function chooses a node to expand from by randomly selecting the search space, X , and selecting the neighbor within the radius, Δ_{BN} , with the lowest cost, $c(v_i)$. If no node exists, the closest node is selected. The

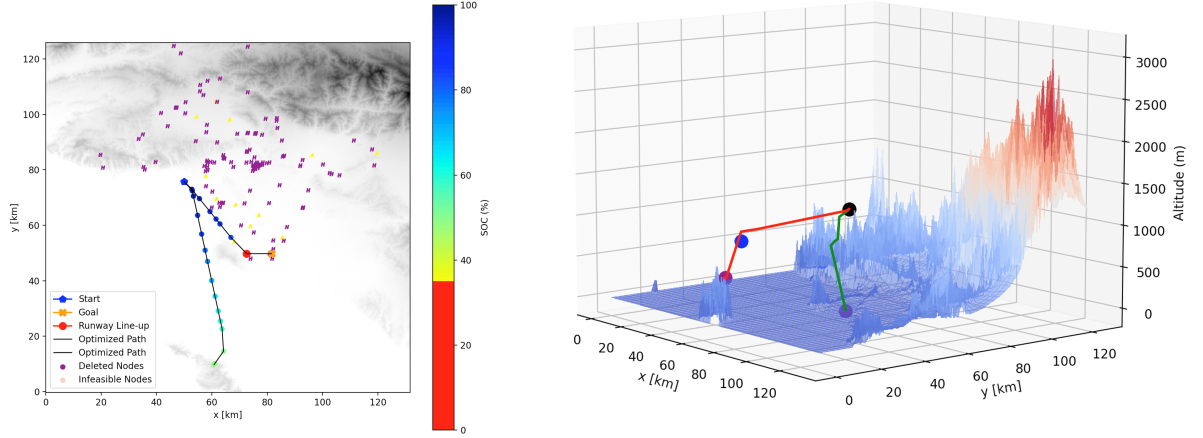


Figure 4.2: Trajectory generation for VTOL vehicles using SST from [131]

next steps are as follows. Next, the forward propagation dynamics model is applied from randomly selected control or motion primitive, MP , sets, U , and a randomly selected time, T_{prop} , using the *MonteCarloProp* function. A new state, x_{new} , is generated that needs to check for feasibility before a new node is stored. The node and edge are stored in the tree if the path between the two nodes is feasible and collision free as reported by the *CollisionFree* function, and the node is locally the best within a distance, Δ_S , to a witness node, $w \in W$, as reported by *LocallyBestNode*. Lastly, the nodes that are marked inactive by *LocallyBestNode*, and are thus dominated by another node in the range of a witness node, are randomly selected to be pruned in *PruneDominatedNodes*.

The *MonteCarloProp* function for the propagation of the random controls along the trajectory holds properties that bound the difference between any two trajectories. The proof states that the worst-case upper bound on the error between trajectories that start at a single node is as in Equation 4.13. This assumes any two controls are such that $\delta u = \sup_t(||v(t) - v'(t)||)$ and the system dynamics hold as Lipschitz continuous with bounded input, bounded output conditions in the small scale. The asymptotically near-optimal property is measured by the bound of the optimal path cost, shown in Equation 4.14

Algorithm 2: Stable-Sparse-RRT($X, U, x_0, T_{prop}, N, \Delta_{BN}, \Delta_S$)

Data: $x_0 \in X, N > 0, \Delta_{BN} + 2\Delta_S \geq \Delta$
Result: $p^* = \{t_1, x_1 | \dots | t_f, x_f\} \in \mathcal{P}$
 $cost(p^*) \leq cost(p_i) \forall p_i \in \mathcal{P};$
 $V_{active} \leftarrow x_0, V_{inactive} \leftarrow \{\};$
 $E \leftarrow \{\}, G = \{V_{active} \cup V_{inactive}, E\};$
 $w_0.x \leftarrow x_0, w_0.p = x_0, W \leftarrow \{w_0\};$
for N **do**
 $x_{selected} \leftarrow BestFirstSelection(X, V_{active}, \Delta_{BN});$ /* Node Selection
 */
 $x_{new} \leftarrow MonteCarloProp(x_{selected}, U, T_{prop});$ /* Node Dynamic
 Propagation */
 $continue;$ /* Feasibility & Neighborhood Checks */
 if $CollisionFree(x_{selected}, x_{new})$ **then**
 if $LocallyBestNode(x_{new}, W, \Delta_S)$ **then**
 $V_{active} \leftarrow V_{active} \cup \{x_{new}\};$
 $E \leftarrow E \cup \{x_{selected}, x_{new}\};$
 $PruneDominatedNodes(x_{new}, V, E);$ /* Node Removal */
 end
 end
end

Table 4.1: SST Parameters

Parameter	Description	Example Value
Propagation Time (s)	Max time for uniform sampling of dynamics propagation	(5.0, 100.0)
Number of Iterations	Base number of iterations to run the SST algorithm	(500, 10000)
Δ_{bn} (Best Node)	The radius for checking for the best node near randomly sample node	(3.0, 25.0)
Δ_s (Best Witness)	The radius for checking for best node around witness nodes	(5.0, 60.0)
Shrink Factor	Factor for reducing the radius size	(0.75, 0.95)
Total SST	Number of times to run SST	(1, 5)

$$||\sigma(T) - \sigma'(T)|| < K_u T e^{K_x T} \delta u \quad (4.13)$$

$$\Delta(c^*) = (1 + \frac{K_c * \delta}{c_\Delta}) \times c^* \geq c^* \quad (4.14)$$

The list of key parameters is found in Table 4.1. These include the propagation time for the forward propagation function, as well as the radius values, Δ 's, for comparing sampled and witness nodes. The addition of the last two parameters, shrink factor and total num-

ber of SST runs, to the SST algorithm defines the asymptotically optimal algorithm: SST* [52]. At each subsequent SST run, the SST* algorithm will increase the number of runs required while shrinking the size of the search Δ . This will leverage the asymptotically optimal behavior of solutions with relatively fast convergence. Decreased values will promote more locally optimal nodes found to propagate from, representing more exploration. An important consideration from the SST theory is that the inequality must hold so that $\Delta_{BN} + 2\Delta_S \geq \Delta$ given the robust clearance, Δ . This also assures that Δ_{BN} balances the exploration and path quality, and Δ_S balances the sparsity and adaptability. The strategy for SST* is to reduce the radii of the Δ parameters, while increasing the number of iterations to find a better path at that size of Δ . This is done by iteratively running the SST algorithm and updating the parameters with the following equations, where ϵ is the shrink factor, N_0 is the base number of iterations, and T is the total number of iterations.

$$\Delta_{bn} = \Delta_{bn} \times \epsilon \quad (4.15)$$

$$\Delta_s = \Delta_s \times \epsilon \quad (4.16)$$

$$N = (1 + \log(j)) \times e^{-(3+1)*j} \times N_0 \quad (4.17)$$

The distance metric used in the algorithm is the 3D euclidean distance. The optimal path cost would be a straight line connecting the start and goal location. The distance metric is also used when comparing the distance of a node to the goal location $d = ||x_i - g||$, for determining whether a path has been formed, and for comparing the distance of nodes to the local witness node, $d = ||x_i - w_j||$.

Table 4.2: k-Nearest Neighbor Experiment for Time Complexity for 100,000 nodes with 90% active, k=20 and r=250

Name	Time Complexity	Processing Time (s)	Query Time K times (s)
Brute Force Array	$\mathcal{O}(n^2)$	0.032	0.381 0.001 (KD)
K-Nearest-Neighbors	$\mathcal{O}(n \log n)$	5.642	0.001

Software Implementation

An additional check is needed to filter out unavailable nodes for constrained nearest neighbor search problems. As later explained, the tree-search algorithm may use boolean equality constraints that must be checked. Therefore, the nearest neighbor software class is developed to check for specific state variables when checking local neighbors. Since this method must be computed many times, the speed of nearest neighbor functions for these large search size problems must be investigated. The brute force can approach the K-d tree performance with the use of a k-nearest neighbor, KNN, structure in which the nodes are sorted and organized as they are compared against the current node in the array.

Table 4.2 shows that the brute force and KNN methods have different performance results, with the brute force approach able to build the tree quicker but is much slower to query for a set of k-nearest-neighbors. This query is repeated many times in the node sampling stage of a sampling-based algorithm. Another selected solution is to use a KNN algorithm to search for the k neighbors in the brute force array nearest neighbors method. Adding a KD-tree data structure for sorting the array can approach the KNN performance without the expensive tree building and rebuilding for problems when applied on the scale of 100,000 nodes.

4.4.2 Dynamic Model

The forward propagation dynamics model is a quadrotor, or quadcopter, that is controlled by two levels of control loops that follow the same translation and rotation kinematics in 3D space. The first model is defined with propellers that are directly controlled by motor rotation rates, ω . The second model plans paths through thrust, T , and attitude targets, (ϕ, θ, ψ) , that are followed with a smooth acceleration-based tracking model. The two are related using the attitude dynamics of a quadrotor that matches the total thrust and body rotation rates assumed from the attitude targets to the four motor rotation speeds. The problem starts with a root finding problem with four unknowns, $\omega_1, \omega_2, \omega_3, \omega_4$, and four equations, F_z^B, L, M, N . The four equations are the body z-axis force, and the roll, pitch, and yaw moments. The aircraft arm lengths, d , yaw drag, e , and moment of inertia, I , are also required. The initial conditions to the root solver, $f(x) = 0$, are the motor speeds for the hovering condition, where $F_z = 0$.

$$\begin{aligned}
 F_1 &= T - k_T * (\omega_1^2 + \omega_2^2 + \omega_3^2 + \omega_4^2) \\
 F_2 &= (\dot{p} - l)I_{xx} - d * (\omega_1^2 - \omega_2^2 - \omega_3^2 + \omega_4^2) \\
 F_3 &= (\dot{q} - m)I_{yy} - d * (\omega_1^2 + \omega_2^2 - \omega_3^2 - \omega_4^2) \\
 F_4 &= (\dot{r} - n)I_{zz} - e * (\omega_1^2 - \omega_2^2 + \omega_3^2 - \omega_4^2)
 \end{aligned} \tag{4.18}$$

The decision is made to create a set of stable maneuver automata to forward propagate the system from one state to another. The setpoint model is selected for experiments in the next section and the parameters selected are seen in Table 4.6. Further details are available in subsection 4.5.1.

Even with accurate modeling of the system and environment, uncertainty causes inaccuracies that may be hard to account for unless it is modeled explicitly. Uncertainty can be split into two categories, aleatoric and epistemic. Aleatoric uncertainty is representative

of random events, while epistemic uncertainty comes from the lack of knowledge of the system or process. The quadcopter dynamics are the most critical item for understanding uncertainty in this work because of both kinds of uncertainty. The two areas of uncertainty are prepared as two uncertainty models for internal acceleration noise and external wind disturbance. The noise causes changes in the acceleration terms of the system using a zero-centered Gaussian distribution, therefore requiring a single parameter, σ_a . The wind model changes the translational velocity of the system using the wind triangle that relates ground speed to airspeed. The wind model, W , is a function of the environment, and the gust model, D , which was detailed in Chapter 3.

The system dynamics are treated as a continuous set of differential equations, as detailed earlier. However, the equations now include additional terms for the uncertainty in the forward propagation. The translational and rotational translation terms in Equation 4.19 now include internal and external noise from the inner loop additive noise, ν , and the environmental wind, W .

$$\ddot{x}_t = f(x_t, u_t, \sigma_a) + W \quad (4.19)$$

Dynamic modeling assumptions have disturbances in the acceleration of the vehicle as shown in Equation 4.20. This could be from sensor noise, control saturation, or model mismatching, and is approximated as a Gaussian random variable, $\sigma_a \sim \mathcal{N}(\mu_a, \sigma_a^2)$. The wind from the static wind field, W , and Dryden gust model, G , affects the dynamics through the wind-axis velocity and the wind triangle, shown in Equation 4.21.

$$\ddot{x}' = \ddot{x} + \sigma_a \quad (4.20)$$

$$v^W = v^B - R_W^B W_{mag} - R_W^B G_{mag} \quad (4.21)$$

4.4.3 Experiments

A simulation environment is prepared that leverages the maps built in Chapter 3, as to compare algorithm performance and discover how the SST algorithm performs for aerial system response. The environment leverages a Python object-oriented approach with the tree-based planning algorithms built using an inheritance class structure, allowing for new algorithms to leverage the functionalities of previous methods. Separate modules for map building and dynamic modeling, which can be accessed through internal class functions, are used to repeat experiments for multiple models and use multiple planners quickly. The urban maps are supplemented with controlled 3D environments of randomly generated or custom obstacle fields. Visualization functions use Plotly [132] and Matplotlib [133].

The environment is prepared to examine two algorithms, RRT and SST. The hypothesis is that SST performs better in the maps of interest, and a series of experiments is prepared to determine if this is true and what are the performance improvements. A preliminary experiment is demonstrated in Figure 4.3 where the RRT and SST have the same dynamic model and objective function. The algorithms are compared for an empty 3D environment of 30 meters in length, width, and height. It is clear that the stochastic tree expansion limits RRT as compared to SST, which has a sparse search expansion. SST is able to find paths quicker and converges to paths that are lower cost than RRT on average. Thanks to the sparse search method, this is done with fewer overall nodes. As a verification, the path is compared against the straight line path between the start and goal points and is within one meter for the euclidean distance measurement.

Stable-Sparse-RRT Performance

The SST algorithm is a powerful tool for finding paths in high-dimensional search space through a sparse search expansion and randomized control actions. The capabilities of SST to find trajectories in 3D space using a quadrotor system are explored through a series of experiments that seek to compare and contrast the algorithm with the vanilla version of the

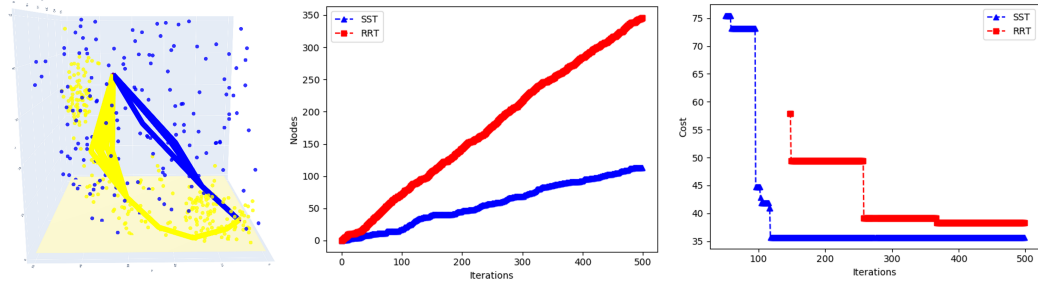


Figure 4.3: RRT and SST Performance in Verification Experiment, RRT nodes and path in yellow and SST nodes and path in blue

precursor, kinodynamic RRT. Parameters are detailed in each section, but some parameters are kept constant throughout the results section unless explicitly mentioned. The constant assumptions include a collision check every 1-meters, dynamic propagation at a timestep of 0.1 seconds, and visualization background processes set to active. The experiments are set up in the framework in a way any future user would be able to replicate, with a predefined set of user inputs, or "knobs," that can be changed to evaluate performance. These "knobs" include the parameters of each algorithm, the wind and noise injection parameters, and the map creation and location.

Comparison to RRT

The SST algorithm is first compared against the baseline kinodynamic RRT algorithm that features the same dynamics and similar propagation parameters. The experiment displays the sparse nature of SST compared to RRT in a random 3D environment with 1000 node checks. The SST algorithm converges on the maximum number of nodes much more quickly than RRT since the witness nodes replace local nodes that are better and prune ones that have a higher cost. In addition, SST Converges to better solutions and finds larger improvements in later iterations as shown in Figure 4.4.

The experiment is repeated over a range of map sizes and with different obstacles to evaluate the distribution of results from the two algorithms. The number of nodes, cost of the final path, and computation time are compared for the two algorithms as shown in

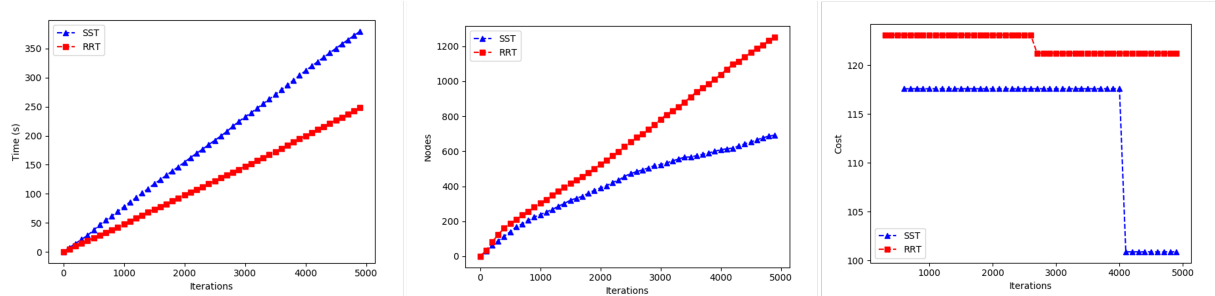


Figure 4.4: RRT vs. SST in Random 3D Environment with Comparisons of Time, Number of Nodes, and Path Cost

Table 4.3: RRT vs. SST in Random Obstacle Maps Performance Comparison

	RRT	SST
Found Path	87%	100%
Lower Cost	42%	58%
Fewer Nodes	13%	87%

Figure 4.5 with RRT in red and SST in blue.

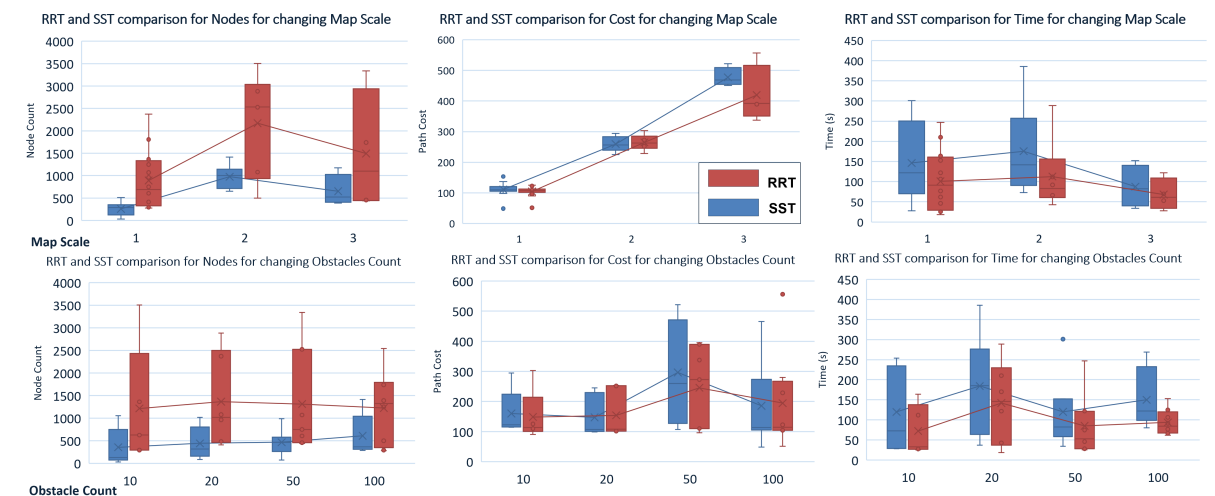


Figure 4.5: RRT vs. SST Multiple Runs in Random Obstacle Maps

Next, a more difficult map is introduced, similar to what was created in the original SST paper by Littlefield [52]. The "hole-in-the-wall" map is used to compare RRT and SST performance in a more complex map, where there is a significant improvement in performance from paths going through the holes in the wall instead of going around the walls. The results are seen in Figure 4.6, where the SST algorithm demonstrates the ability to use

51% fewer nodes while having a 13% higher chance of finding a path but at a 16% higher cost when using these parameters. However, if the parameters are changed by reducing the bounds of the propagation time parameters and reducing the values of Δ_{BN} and Δ_S , then the performance is improved. A couple of outlier cases from the 2000 iteration constraint keep the SST performance from being even better. Comparing the algorithms on a case-by-case basis results in SST having a lower cost path 61% of the time, using 31% fewer nodes and converging to the solution 50% faster on average. The converged paths are only 10% better on average, but this is to be expected for the finite-time performance of SST with a single set of parameters. The SST* algorithm, which guarantees asymptotically near-optimal paths, can be leveraged when longer runtimes are available.

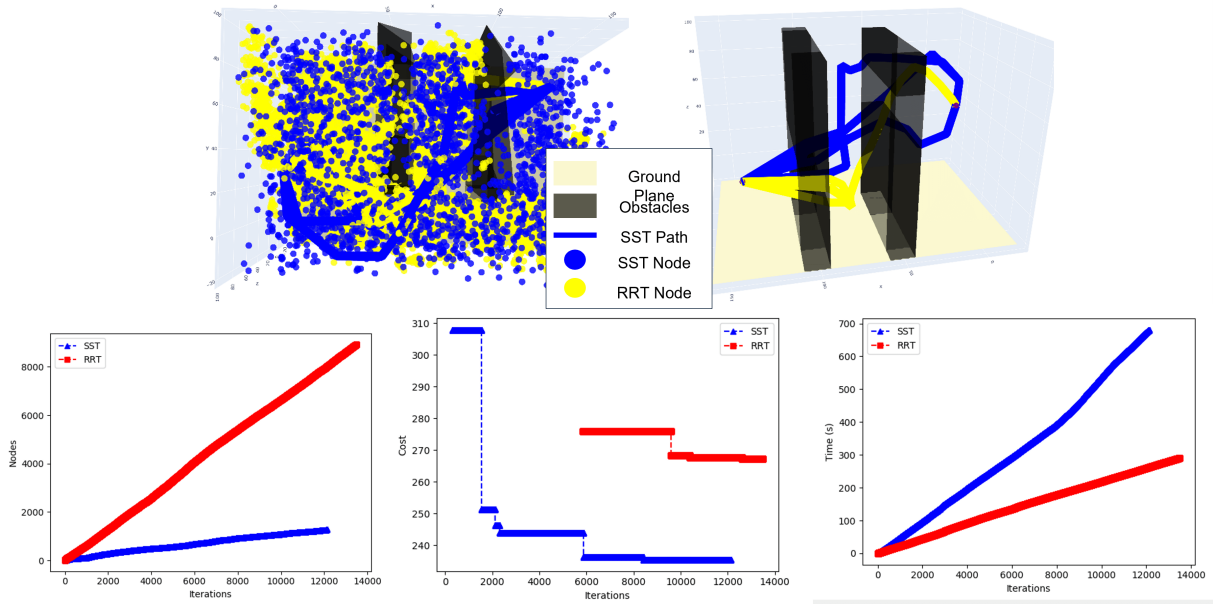


Figure 4.6: RRT vs. SST Hole in the Walls

Application in Urban Maps

Now, urban maps are used to test the ability for the algorithm over large distances with terrain and structure obstacles. The performance of RRT and SST on two urban maps generated from Chapter 3 is shown in Figure 4.7 and Table 4.4, where the SST algorithm demonstrates its sparsity and consistency. The experiment was run with limits on the algo-

Table 4.4: RRT and SST Solution Probability for 2000 iterations for Two Urban Maps Locations

	RRT	SST	Change
LA	33%	93%	60%
NYC	47%	100%	53%

rithm iterations but also for the SST algorithm to use the same number of nodes. Therefore, demonstrating the improved ability to find trajectories over a large search space with the same or fewer nodes compared to RRT.

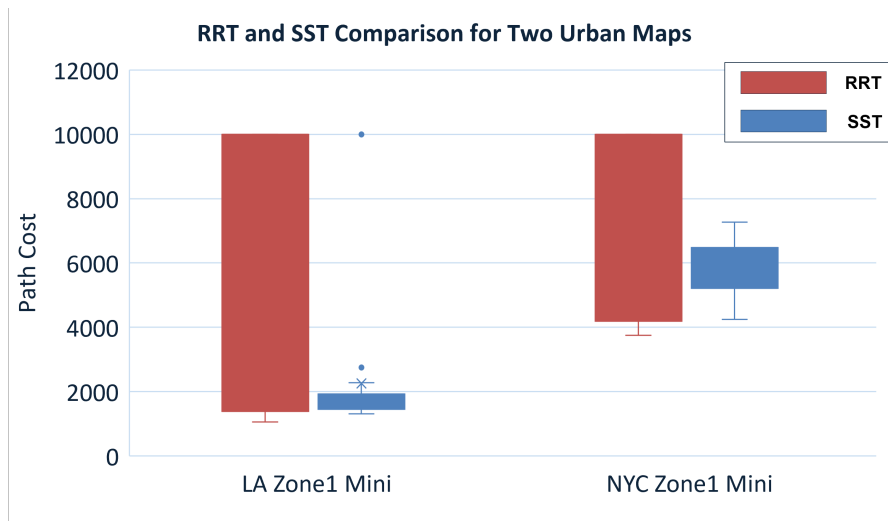


Figure 4.7: RRT vs. SST Comparison for Two Urban Map Locations

Figure 4.8 shows the final trajectories in demonstrations for each map. The results show how trajectories are formed over terrain changes and around building obstacles. The primary benefits of SST over large urban maps are shown to be sparsity and consistency. RRT can be defined as more random, causing the algorithm to get caught in poor search spaces and need more memory. Furthermore, the SST algorithm proves the capability of sampling-based planners that was hypothesized. The algorithm is able to find trajectories that change in altitude by climbing over terrain and that aggressively maneuver around obstacles by dodging buildings.

The time to compute the flights is expensive because of the quadcopter dynamics model requirements to calculate the smooth flight path. This is similar to the issue in previous

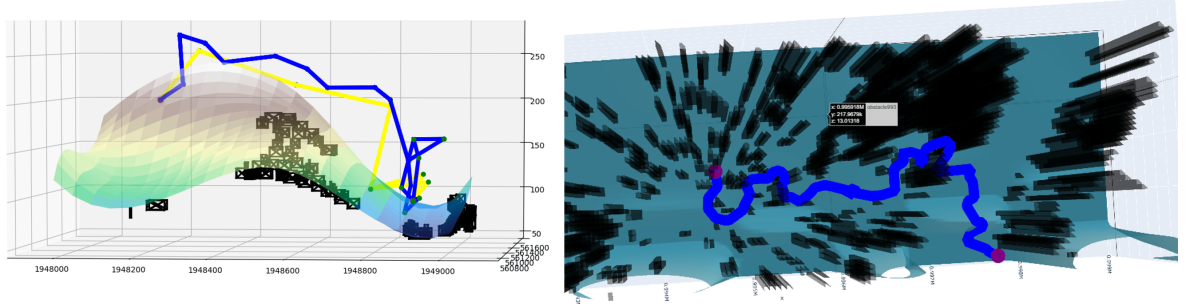


Figure 4.8: RRT vs. SST Trajectories through Terrain and Structures for Los Angeles (Left) and New York City (Right)

works [130, 131]. The assumption was made that the dynamics model would account for accurate flight paths over long distances and calculate energy metrics while being impacted by environmental and system uncertainties. Therefore, solutions like linearization or response surface models are not leveraged in this work. Instead, motion primitives are explored that can perform smooth flight paths quicker while improving the planning algorithm by selecting only relevant flight maneuvers. It is hypothesized that this will lead to a speed improvement and convergence of better paths in finite-time limits.

4.5 Motion Primitive Formation for Efficient Planning

Frazzoli in [134] describes how the path planning problem tends to grow exponentially with respect to the dimension of the configuration space. The configuration space could grow rather large for complex and high-dimensional dynamic systems. One solution to the curse of dimensionality occurring in complex dynamics trajectory planning was demonstrated by the maneuver automaton as detailed by Frazzoli in [134] and explained in more detail by Lavalley [39]. The maneuver automaton, alternatively defined and named motion primitives, can provide efficient queries of kinodynamically constrained paths. Previous works have approached this by simplifying the dynamics for point-to-point navigation, as by Mueller in [135]. Other approaches, such as [136], solve a discrete set of trajectory problems using an optimal control formulation, which is then sampled during planning. This idea has been examined as relates to lattice-based motion planning, where a discrete set of forward

propagation states are selected for planning.

Finding trim states for maneuver automata has been employed in the past for complex aerial vehicles, such as by Schouwenaars et al in [137]. The methodology sought to compute offline and open-loop trajectories that generates a maneuver library. The final result was a set of 48 trimmed maneuvers that had been computed with external uncertainties included while solving for the open-loop trajectories.

This work leverages this idea to build a set of open-loop trajectories by querying the quadcopter model along promising vehicle maneuvers in an offline simulation framework and then improving the quality of the flight path of an open-loop lower-order dynamics model. First, the initial trajectory is flown by the acceleration-based quadcopter tracking model. Next, the lower-order motor speed controls are solved using the kinodynamic equations and attitude dynamics and found by use of a root solver matching the forces and moments. Next, the trajectory is iteratively improved using the Differential Dynamic Programming (DDP) algorithm. The result is a lattice-based motion primitive set of climbs, turns, cruise, and hover for a quadcopter.

4.5.1 Dynamic Models

The UAS model of interest is a quadcopter, as seen in Figure 4.9, and is a slightly modified version of the work from [138]. The world reference frame \mathbf{W} and the body fixed frame \mathbf{B} are used for defining the system dynamics and are related through the rotation matrix R_{WB} . The body forces acting on the system can be described proportionally to the square of the rotation speed of the propellers, ω^2 , by the constant k_T . The body moments, M, L, and N, can be described similarly. The equations are detailed below using $i \in [1, 4]$ to denote the different motor and rotor pairings. The thrust constant is defined as k_T and the moment constant as k_τ .

$$F_i = k_T \omega_i^2 \tag{4.22}$$

$$M_i = k_\tau \omega_i^2 * (-1)^i \quad (4.23)$$

The angular rates, Ω , and moment of inertias, J , describe the attitude dynamics. The aerodynamic forces, F_a , on the body can then be defined as proportional to the force of each rotor by the factor k_D . External disturbances can be represented by F_e . A subset of the resulting equations of motion are shown in the following equations.

$$\dot{x} = v \quad (4.24)$$

$$\dot{v} = \frac{1}{m}(R_{WB}F_T - R_{WB}F_a + F_e) \quad (4.25)$$

$$\dot{R}_{WB} = R_{WB}\Omega \quad (4.26)$$

$$R_{WB} = \begin{bmatrix} \cos(\alpha)\cos(\beta) & \sin(\beta) & \sin(\alpha)\cos(\beta) \\ -\cos(\alpha)\sin(\beta) & \cos(\beta) & -\sin(\alpha)\sin(\beta) \\ -\sin(\alpha) & 0 & \cos(\alpha) \end{bmatrix} \quad (4.27)$$

$$J\dot{\Omega} = -\Omega \times J\Omega + A \begin{bmatrix} \omega_1^2 \\ \omega_2^2 \\ \omega_3^2 \\ \omega_4^2 \end{bmatrix} \quad (4.28)$$

The dynamics model must account for the inner loop dynamics for trajectory tracking. Therefore, a model of the low-level attitude controller is constructed with the desired angular rate of the system as the input, and the gain, k , and time constant, τ , as the tuning parameters. The model was tested in flight simulation for closed-loop feedback control to tune the parameters to the desired open-loop performance. Experiments show that bounded

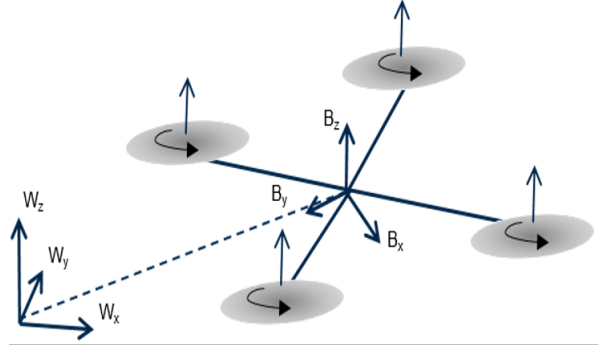


Figure 4.9: Quadcopter Model for Flight Simulation

$\dot{\theta}$ and $\dot{\phi}$ values when tracking targets keep the system in stable flight conditions while also improving the reaction time of quick maneuvers.

$$\dot{\phi} = \frac{1}{\tau_{\phi}}(k_{\phi}\phi_{cmd} - \phi_{ref}) \quad (4.29)$$

$$\dot{\theta} = \frac{1}{\tau_{\theta}}(k_{\theta}\theta_{cmd} - \theta_{ref}) \quad (4.30)$$

Drag on the aircraft is taken to be a force that is defined by the Raleigh drag equation in Equation 4.31. The constants for drag, c_d and the aerodynamic lift area, A , are selected from the literature, while the temporal parameters like density, ρ , and velocity, v , are calculated during the state transition.

$$D = \frac{1}{2}\rho v^2 c_d A \quad (4.31)$$

The energy consumption can be modeled in many different ways. There exists literature that has explored this for quadcopters and VTOL, for example, a first principles approach by Rodrigues in [139, 140]. A further expanded calculation of power in multiple flight modes is conducted by Stolaroff in [141]. Many of these models stem from the key paper for quadrotor helicopter dynamics and power calculation by Hoffmann in [142].

Quadcopter dynamics results from using formulas from quadcopter energy modeling.

Assuming a forward flight, the model is adapted from Hoffmann in [142] and is detailed in [139, 140]. Alternative techniques attempt to model from data. For example, in [66], data on a quadcopter is used to train a deep neural network to estimate the energy use over the flight in an accurate and computationally efficient manner. However, without direct access to accurate and informative data, these methods are unsuccessful. Therefore, the Hoffman model from [142], is used. The power calculation relies on an estimate of the induced velocity, v_i , and then solving the implicit equation at different flight conditions. The efficiency term, μ , is used to tune the calculation for the model of interest.

$$P_{min} = T(v * \sin(\alpha) + v_i) \quad (4.32)$$

$$v_i = \frac{2T}{\pi n D^2 \rho \sqrt{(v \cos(\alpha))^2 + (v \sin \alpha + v_i)^2}} \quad (4.33)$$

$$P = P_{min} / \mu \quad (4.34)$$

The M100 DJI Matrice is selected to model because of the availability of data and previous experiments. The model is defined with parameters found in [143] or estimated from information in the DJI website, shown in Table 4.5. The energy model is validated against data from [144], with an example of flight energy data seen in Figure 4.10. The quadcopter model's hover power output has a mean of 456 W and a standard deviation of 16, while the flight data has a mean of 459 W and a standard deviation of 28. The mean power error is only 0.66% and is deemed to be acceptable for the limited maneuver set of interest.

Table 4.5: DJI M100 Parameters

Mass, m (kg)	Drag Coefficient, Cd	Moment Inertias Ixx, Iyy	Thrust constant k _T	Drag constant, K _d	Battery Mass Ratio	Battery Energy Density
3.71	0.65	0.155, 0.166	1.85E-5	1.0E-2	0.15	200

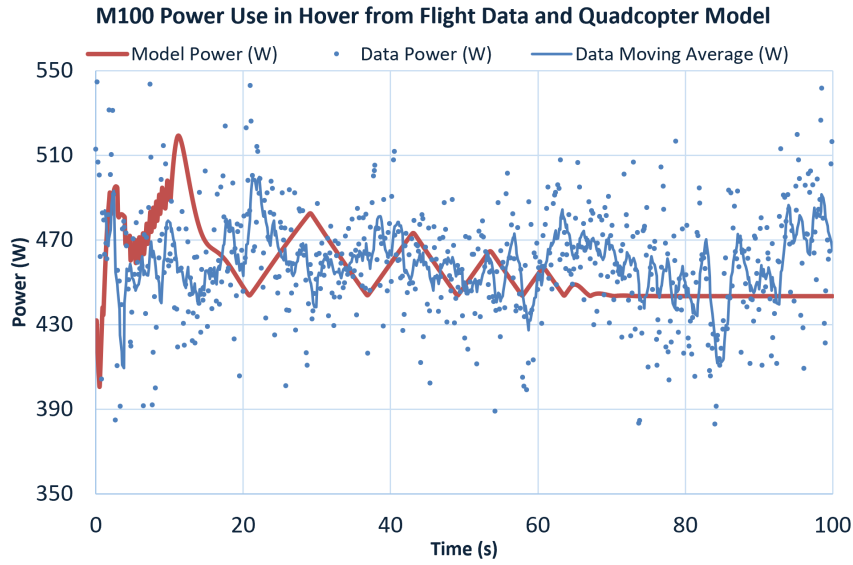


Figure 4.10: DJI M100 Hover Data for Quadcopter Model compared to Experimental Data

4.5.2 Stable Maneuver Automaton

Improved dynamic modeling for kinodynamic constrained trajectory planning requires a maneuver automaton framework to compute offline open-loop trajectories rapidly and efficiently. The initial maneuvers that are used in the previous section's results are the uniformly randomly sampled controls in terms of roll, pitch, yaw, and thrust force that keep the vehicle capable of closed-loop stability. The values for the parameters are seen in Table 4.6.

Table 4.6: Random Control Target Motion Primitive Parameters

Dynamics Setup	Heading Command	Roll Command	Pitch Command	Thrust Command	Times (s)
<i>Min</i>	0	-2	-2	34.29	5
<i>Max</i>	360	2	2	37.90	20

The dynamic system is defined as a continuous set of differential equations but is solved in discretized time steps. Therefore, the difference between each time step, ΔT , must be selected as a tradeoff between accuracy and time. In Figure 4.11, the time steps of 0.01, 0.10, 0.20, and 0.25 are compared for 5 random motion primitive selections. Values equal to or greater than 0.30 resulted in large errors that were not beneficial to show. Each time

step and motion primitive combo was repeated five times to get a quick distribution of results under windy and noisy flights. Overall, the time step selections here were able to keep flight path errors within a few meters along the entire flight, and energy estimates within 0.5 Watt-hours, W-Hr. The runtime increases for smaller time step values but is not dramatically different except for the 0.01 value.

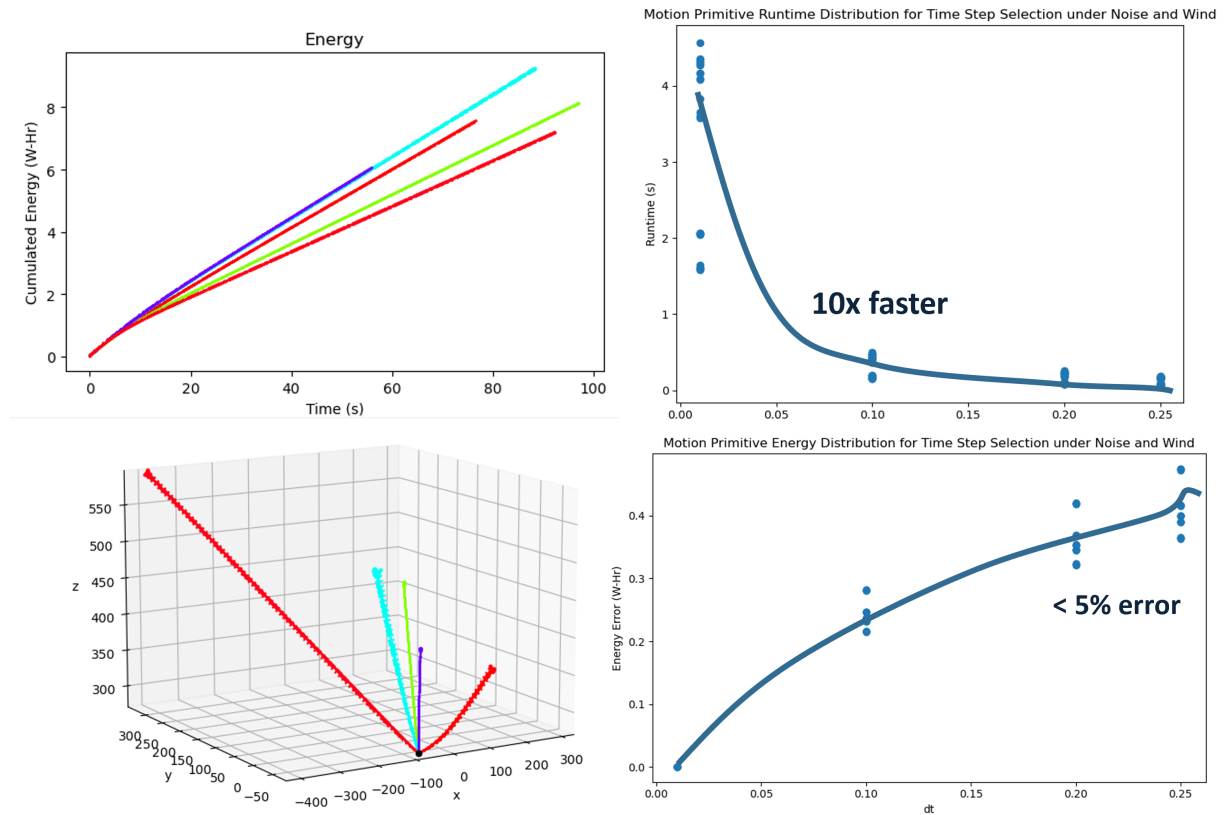


Figure 4.11: Energy, Flight Path, and Runtime Performance for 5 Motion Primitives with Different Time Step (dt) Values

The equations from LQR theory found in section A.4 are used to evaluate the existence of closed-loop stable paths for motion primitive paths, in particular when there is wind or noise in the dynamics. This is important as the tracked paths will be achieved through a closed-loop controller when the mission is attempted. Therefore, if the open-loop controls can be improved through closed-loop control, the assumption of the open-loop path being a lower-bound estimate of performance is achieved.

Figure 4.12 shows the motion primitives tested for flight stability. Trajectories in green

are open-loop stable and retained within the motion primitive set. The paths in yellow are not open-loop stable but are closed-loop stable. These are filtered out of the motion primitive set in use, for now, but confirm the hypothesis that all trajectories within the frames of the motion primitive parameters are closed-loop stable.

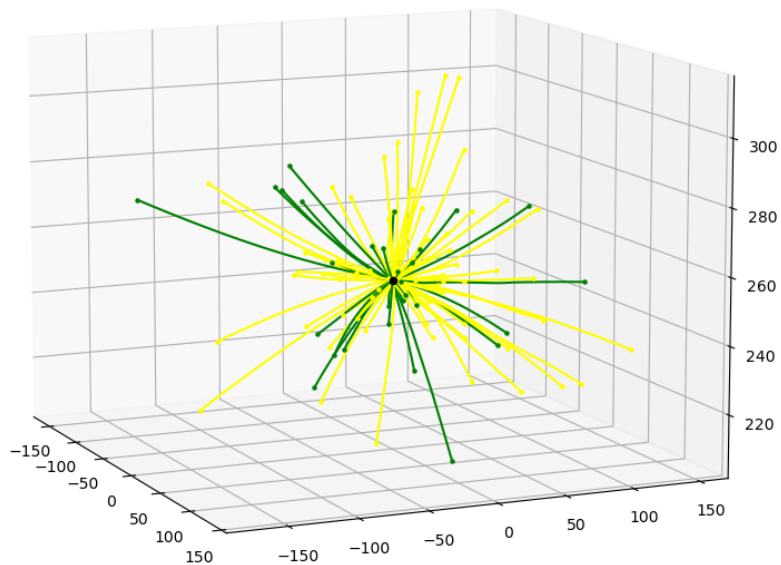


Figure 4.12: Motion Primitives Indicating Open Loop (Green) or Closed Loop (Yellow) Stability

4.5.3 Learning Motion Primitives

Examples using motion primitives or accurate steering functions from two states are extensive in literature. For example, work such as by Mueller in [135] used closed-form equations that can be quickly solved and are generated for specific vehicle primitives, which have been prepared to minimize an aggressiveness cost function. In [134], motion primitives are defined as time-parameterized curves that form a *maneuver library* for the system dynamics of interest. These primitives can be interconnected to form complete, feasible, and optimal trajectories while reducing the computational complexity of the solution that is often referred to as lattice-based planning.

Much work has investigated motion primitives for constrained path planning or kinody-

dynamic motion planning. Work such as [145] and [146] leverage motion primitives to search directly in a state lattice. It is common to solve the motion primitive problem using optimal control methods, in particular, a two-point boundary value problem (TPBVP). Another method developed an efficient trajectory library using filtering by Viswanathan in [147].

Results for the motion primitives involved setting up the rapid quadcopter point-to-point model and then leveraging Differential Dynamic Programming to improve the accuracy and robustness of the trajectory. This allows handling any black-box dynamics model with any parameters to create a motion primitive set. A Latin hypercube sampling method is used to choose the state targets that mark the primitives because the Latin hypercube sampling should produce an even distribution across the state space.

Differential Dynamic Programming

DDP, or Differential Dynamic Programming, is selected as it has shown success in cases with nonlinear dynamics that are difficult to linearize, such as the inverted pendulum problem. DDP is described in detail in [148] and has proven successful in similar applications from a previous work by Hyunki and Harris [149], where the algorithm is used to produce energy-optimal trajectories for hybrid VTOL aircraft. DDP utilizes local dynamic programming methods and a quadratic approximation of the Q function. A pseudo-hamilton is formed from the second-order expansion of the Q-function. Also, the optimal control is defined as a perturbation from a nominal control.

DDP can be referenced as a shooting method as it relates to optimal control. The control and state are always feasible and the problem is solved using unconstrained optimization. More specifically, DDP iteratively updates on a nominal control set by solving forward passes of the dynamics and objective function, calculating second-order approximations to the state and control dynamics about each discrete time step, and then backpropagating an update function. The updates culminate in Equation 4.35b, which is effectively a line search in the optimal control space. The parameter γ is the learning rate, k_i is the feedback

gain, K_i is the feed-forward gains, x is the state vector, and u is the input vector. k_i and K_i are determined by the quadratic expansion of the objective function. The controls during the forward propagation are bound with soft constraints through a squashing function seen in Equation 4.36a and detailed in [148]. The algorithm is summarized in algorithm 3.

$$\delta u = k_i + K_i \delta x \quad (4.35a)$$

$$u_{i+1} = u_i + \gamma \delta u \quad (4.35b)$$

$$x_{i+1} = f(x_i, g(u_i)) \quad (4.36a)$$

$$g(u) = \frac{b_u - b_l}{2} \tanh(u) + \frac{b_u + b_l}{2} \quad (4.36b)$$

Algorithm 3: DDP(X, x_0, N, γ)

Data: Input: $\bar{x}_0, x_f, \bar{u}, \gamma, Q_f, Q, R$.

Result: Output: J^*, u^*

Init($\bar{x}_0, x_f, \bar{u}, \gamma, Q_f, Q, R$) ;

while $|J_i - J_{i-1}| > \epsilon$ **do**

 forward propagation of the dynamics with u_i, A_i, B_i ;

 backward propagation of quadratic expansion of the objective function

$Q_o, Q_x, Q_u, Q_{xx}, Q_{uu}, Q_{ux}$;

 state propagation from quadratic expansion of the objective function

$u_{i+1} = u_i + \gamma \delta u$;

 calculation of new objective function $J(u_{i+1})$;

end

The objective function is defined as in previous literature as Equation 4.37, where x_N and x_t are the states at the final time step, N , and every other time step, t . x_f and x_n are the desired final state, f , and the nominal states from a reference trajectory formed from the nominal control u_n . u_t are the applied controls at time step t , which will update from the previous line search equations. The matrices Q_f, Q , and R act as the weight parameters

for the quadratic cost function, and apply to the final state, path state, and control inputs, respectively.

$$J = (x_N - x_f)^T Q_f (x_N - x_f) + \sum_{t=0}^{N-1} (x_t - x_n)^T Q (x_t - x_n) + u_t^T R u_t \quad (4.37)$$

4.5.4 Experiments

The nominal control is formed using the inverse attitude dynamics root solver introduced earlier. A point-to-point tracking planner that seeks to minimize acceleration errors is used to form a path of thrust and angular setpoints, which was defined as the output of the quadcopter model. The inverse model estimates the quadcopter motor speeds, ω_i , to achieve the forces and moments over time. Any error in the inverse solution is expected to be fixed during the DDP algorithm. The state cost, Q , and control cost, R , are normalized from 0 to 1 combined before passing into the DDP cost function. The DDP algorithm iteratively solves for trajectories that minimize the quadratic cost function. An example of the results are seen in Figure 4.13. Trajectories are improved by reducing the final position error by 15% and reducing the energy use by 5% on average.

The selected states and the resulting trajectories are seen as a lattice of maneuvers in Figure 4.14. The lattice of maneuvers is accessible as a set of motor speeds, $\omega_{\{1,2,3,4\}}$, and can be queried during the sampling-based planner expansion. Figure 4.15 shows the maneuvers selected for two maps. The results are produced much quicker, with a reduction of trajectory iterations from an average of 2.15 seconds to 0.15 seconds, a speed increase of 13 times. The maneuvers are smooth enough that they can be limited for any time from T_0 to T_f and are assumed to be easily stabilized. Figure 4.14 shows the flight patterns limited by the minimum time step used in the SST algorithm.

The control actions, or motion primitives, are demonstrated in both generated and urban maps in large-scale map sparsity and solution quality with finite-time runs. This is because

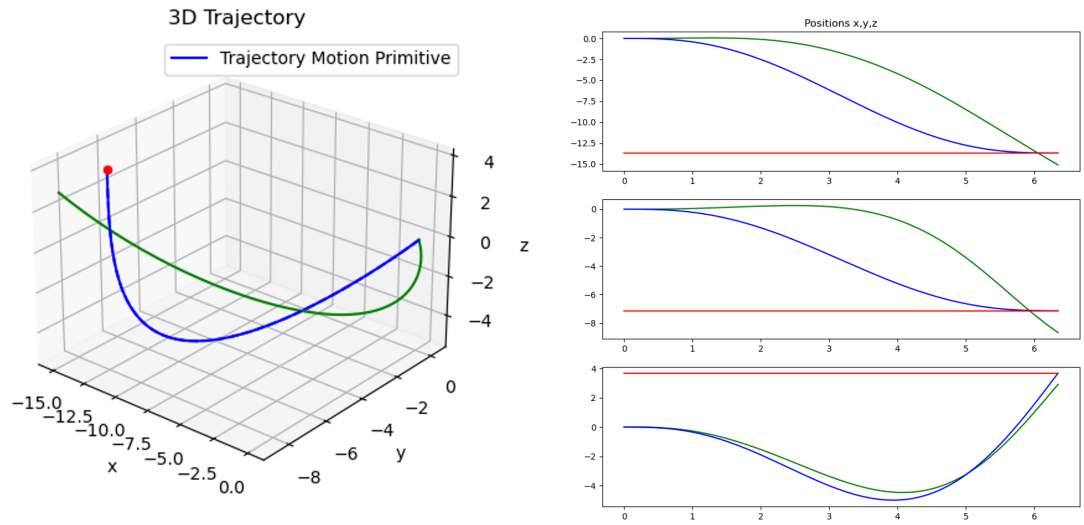


Figure 4.13: Motion Primitive Solution showing the Original Trajectory in Green and DDP Improved Trajectory in Blue

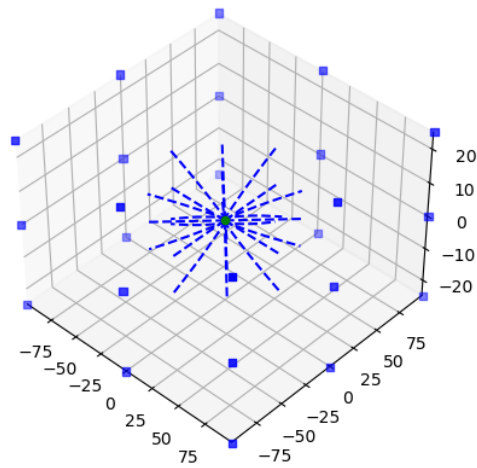


Figure 4.14: Motion Primitive Lattice

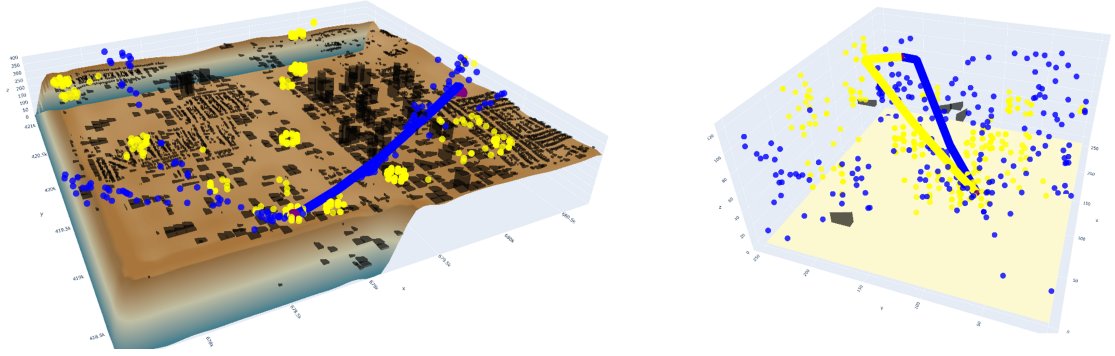


Figure 4.15: Empty Map and Atlanta Urban Map Trajectory Results using Motion Primitive Control Actions

of the consolidation of states in the lattice shape and limited distance covered by the lattice motion primitives that always return to a hover state. In small-scale maps, this is much more successful than in large-scale maps. Therefore, improvements to create motion primitives for longer flight maneuvers would be needed for further use in urban map settings.

4.6 Summary and Future Work

The sampling-based planning algorithm SST is combined with a forward propagation dynamics model and compared to the kinodynamic RRT algorithm. SST performs better across the random 3D maps and the urban maps created from the previous chapter. Time constraints lead to the creation of random motion primitives for an acceleration-based tracking model to be replaced with direct motor speed values that are learned for a lattice primitive creation algorithm. The algorithm generates nominal flight paths and improves them using the DDP algorithm.

Speed improvements could be achieved by learning the power or energy for the dynamics at each time step. The energy could be predicted using publicly available data for the M100 as in [144] and using surrogate models as in [66]. In addition, adding reachability constraints can provide further guarantees in the dynamic propagation model that the motion primitives used are safe and stable. More discussion and background is available in

section A.4.

The two experiments provide the ability for an aerial system model to efficiently explore 3D maps and produce flight trajectories with both attitude and thrust targets, as well as direct motor speed values, in noisy and windy conditions. Furthermore, the algorithms are proven to be successful in large maps with obstructions.

There still exists concerns for the energy along the flight as well as the situational awareness outcomes of the 3D trajectories. Therefore, further investigation is needed into the commonly referred to area of research, risk-aware planning. The next chapter introduces this concept and explores a novel approach to modeling and optimizing a risk-aware planning algorithm for aerial situation awareness in disaster response.

RISK-AWARE PLANNING WITH INFORMED SPARSE ROUTING

5.1 Introduction

Extensive work is ongoing for applying ground or aerial robots to complex environments like vehicle racing¹ or underground caves². However, the application of urban disaster environments is still underdeveloped, especially at this time of increased disaster frequency and growth of interest from private companies like Brinc³ and Skydio⁴. For example, recent work by Best et al. in [150] investigated multi-robot systems in complex, multifarious environments. This work leveraged a global planner of RRT-Connect with vision-based viewpoint states that seek to visualize new areas in the environment. Alternatively, low-altitude urban planning was investigated for the best metrics by Ochoa in [65]. However, this work was focused on 2D planning and not focused on disaster response.

Search and rescue operations in the setting of a natural disaster allow different vantage points at low altitudes, as shown by Scherer in [35], where generating the safest possible paths is not trivial. The concept of aerial flight safety is explored in detail by Donato in [37]. Furthermore, "additional research is needed to incorporate risk metrics for urban flight planning, such as system, actuator, sensor, and weather-related risks, to extend current fixed-altitude maps to full 3D cost maps to support full 3D flight planning" as explained by Ochoa in [49]. Ochoa and Atkins in [65] define the risk or cost of flight through map-based and path-based metrics for urban flight planning of UAS. However, this work focuses on static risk metrics that do not consider the vehicle, environmental changes, and uncertainties during a disaster or SAR event.

¹<https://www.lockheedmartin.com/en-us/news/events/ai-innovation-challenge.html>

²<https://www.darpa.mil/program/darpa-subterranean-challenge>

³<https://brincdrones.com/>

⁴<https://www.skydio.com/public-safety>

This chapter presents a novel method to leverage the uncertainty of the environment and dynamic models to predict the risk of mission failure along a flight trajectory and find optimal flight paths for visual data collection. The approach includes investigating the algorithms and models for predicting outcomes of risk metrics along a flight trajectory while leveraging the planner from Chapter 4.

5.2 Background

The primary objective is to find a feasible set of configurations from the start to the goal location. Flight planning requires a series of decisions to perform the mission optimally. For one, the data of interest can dramatically change the type of processing and the method to solve the problem. Next, some form of costmap is necessary when exploring the environment. The costmap may be as simple as an occupancy grid to prevent obstacles but could be more advanced and represent the probability of failure for the mission, which we will call a *riskmap*. Next, an algorithm is required to solve the problem, avoid an exhaustive search and find the *best* solution as soon as possible with guarantees. Lastly, an objective is necessary to decide what makes a path optimal. For instance, the RRT algorithm and A* algorithm will traditionally seek to minimize the Euclidean distance, which is a first approximation to the true objective, for example, total energy use.

Xiao, Dufek, and Murphy formally defined risk as the probability of not completing the path, or in other words, mission failure [151]. Xiao continued this in [62] by formalizing risk metrics to distinguish conditional probabilities of mission failure, for example, collisions and energy, that unifies all adverse effects into a single numerical metric. Further details are found in [152]. Previous risk metrics have been defined as the probability of collision, control failure, or loss of localization but not as a formal representation of mission failure. Previous works by Di Donato [59], and Harmsel [60] applied risk prediction to rotorcraft emergency landing. A safe landing zone and flight path were required to provide confidence to rotorcraft operating in crowded environments. Graph search algorithms

that accounted for dynamic constraints were used to plan feasible flight paths. The target locations for the flight path were chosen by using offline data on the ground terrain and population density. Therefore, proven examples have shown how costmap formation and flight planning can be used for aerial navigation and decision-making.

An alternative strategy to flight metrics has been to create risk maps, or costmaps, of the environment that can be computed before flight and updated with in-flight metrics. Combining datasets can provide richer features or improve the robustness of noise in the decision-making process. Fusion, if done correctly, increases the amount of actionable intelligence. For example, urban segmentation is performed by Montoya-Zegarra et al. in [153], using a three-step process: context-aware classification, hypotheses generation, and inference. First aerial data provides predictions on buildings and roads, and then a digital surface model is used to improve candidates through matching and pruning. The result is labeled asphalt, roads, trees, and grass. Similarly, Singh et al. in [154] combined geometric and semantic landing zone evaluations by filtering onboard low-resolution lidar and Support Vector Machine training with fly-over high-resolution lidar, respectively.

The methods of preparing the dataset and combining them are what make a costmap formation algorithm. Consider Figure 5.1, where terrain and population information is combined into a *riskmap*. Often, these datasets are not compatible. For example, one may be raster data, while another is vector data. The resolution and uncertainty of the data will likely not be the same. For instance, census tracts often represent population density data, while terrain classification or slope could be at the resolution of inches from lidar or visual data.

Temporal or dynamic data can also be considered, adding another dimension to the costmap. Another step of prediction or estimation may be required now that the labels are not independent of another external variable. An example would be weather data used during flight planning, as by Kim in [155], which may lead to a database query or a weather forecasting model request.

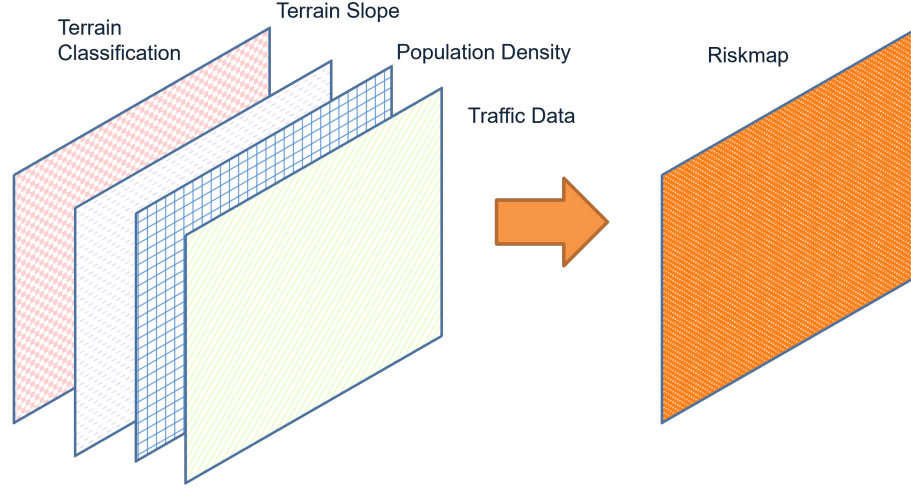


Figure 5.1: Combining Data Layers for Riskmap or Costmap in [48]

Many works in recent years have taken advantage of the safety guarantees provided by CVaR constraints because of the fact that, unlike chance constraints, CVaR constraints are coherent, convex, and distinguish between tail events. In [66], Choudhry used CVaR to measure the risk of a distribution of paths from Monte Carlo simulation, and in [63], Fan used Monte Carlo simulation to learn the CVaR metric for use as a risk assessment. In [69] the safety risk measures the conditional expectation of the distance between the robot position $y(t)$ and the safe region. Therefore, CVaR is a proven technique for using statistical data to measure mission risk through the tail of the distribution, or worst-case events.

5.3 Research Question 3

The framework addresses flight safety when applied to offline flight planning for disaster response. The response during field robotics applications requires safe flight decisions even when the environment and system have large amounts of uncertainty. In addition, the framework seeks to address the gap in modeling and measuring situational awareness to improve flight patterns. The research question is considered as follows.

Research Question 3

How can risk measures be incorporated into trajectory planning algorithms to gain confidence for safe and informative flight in uncertain environments?

The hypothesis is that a formal analysis of the most critical risk metrics and the use of a single statistical measure of risk in the planning algorithm can measure the safety of flight maneuvers, therefore leading to more robust and informative trajectories. The assumption is that all risk measures can be defined entirely through the energy used in flight, the distance to collision with an obstacle, and the data collected for first responders at special *landmarks*. Furthermore, the methodology seeks to provide confidence in mission success by finding improved paths with faster convergence and better finite-time performance.

The experiment is divided into two tasks to investigate the development of a risk-aware sampling-based planner and the efficiency and accuracy of risk metrics in formal reasoning. The experiment formally defines and models the risk metrics that should be used for aerial flight for situational awareness. The risk models are evaluated on how they can be used in the risk-aware planning algorithm and to determine the most informative modeling approach for each. Then, the experiment develops and improves the risk-aware planning algorithm using advancements to the sampling-based planner from Chapter 4.

5.4 Risk Reasoning for Safe Flight

Previous methods for path planning often employ the classic quote, "optimism in the face of uncertainty." This, however, can be dangerous when faced with many unknowns and uncertainties during flight. Therefore, it is best to account for as many of the unknowns as possible. This is the foundation of why risk-aware planning is utilized.

Risk is considered to be a functional, \mathcal{F} , which takes as input the risk metric function, \mathcal{R} , and the risk mapping function, \mathcal{M} , and transforms the state, environment, and other inputs, E , into a risk vector, C in configuration space.

Table 5.1: Definitions of Risk Terms

Risk	A cost or constraint considering probabilistic uncertainties
Risk Map	Data structure to store risk values in appropriate subspace
Risk Metric	Mapping between system state or actions to risk values
Risk Value	Used to compare states or nodes of a path

$$E \in R^k \xrightarrow{\mathcal{F}(\mathcal{R}, \mathcal{M})} \mathcal{C} \in R^d$$

The risk metric function is defined to solve for a risk vector in risk space, R^r , and the risk mapping function transforms the vector from risk space into configuration space. Previous methods include Gaussian Process and Factor Graphs as in [156] and [157]. The four key definitions for risk reasoning and risk-aware planning are seen in Table 5.1.

5.4.1 Risk Under Uncertainty

Conditional-Value-at-Risk (CVaR) is a coherent, convex metric and can distinguish between tail events better than chance constraints [70]. CVaR, seen in Figure 5.2, has traditionally been used in risk management for financial planning; however, in recent years has been adopted for risk-aware planning for dynamic systems. In [158], Sharma et al. leverage the Conditional Value at Risk to find risk-aware paths, where the goal was to maximize the CVaR of travel efficiency or the inverse of travel failure. CVaR is a coherent and averse risk measure that can be expressed as a minimization formula and preserves convexity as explained in [70]. Background information on probability theory, Gaussian random variables, and more are found in section A.1.

The derivation of CVaR starts with a random variable X , which is used to represent some concept of *loss* or *risk*. VaR, for a confidence bound α , is seen in Equation 5.1. The formula from Rockafellar [159] is seen in Equation 5.2.

$$VaR_\alpha(X) = \min \{z \mid F_X(z) \geq \alpha\} \quad \forall \alpha \in [0, 1] \quad (5.1)$$

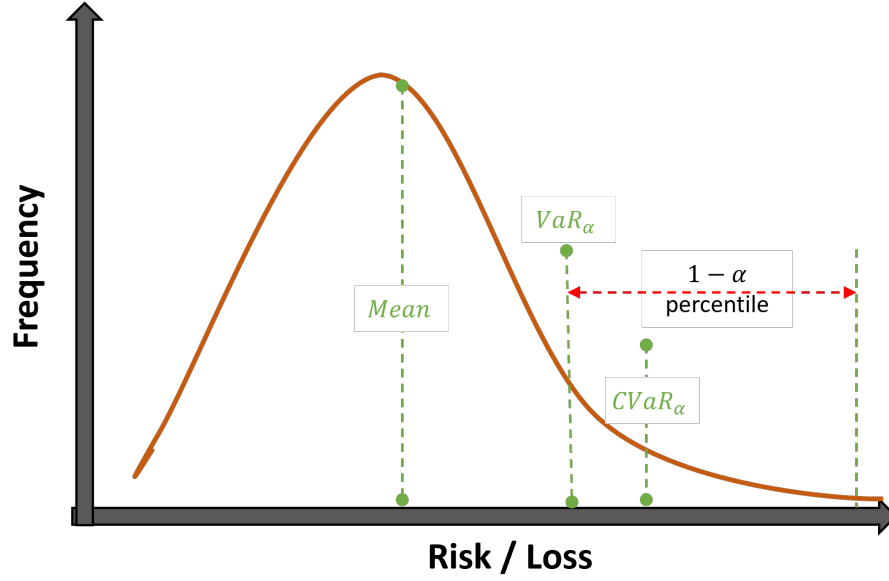


Figure 5.2: Conditional Value at Risk for Distributions with percentile α

$$CVaR_\alpha(X) = \int_{-\infty}^{\infty} z dF_X^\alpha(z) \quad (5.2)$$

$$F_X^\alpha(z) = \begin{cases} 0 & \text{when } z < VaR_\alpha(X) \\ \frac{F_X(z) - \alpha}{1 - \alpha} & \text{when } z \geq VaR_\alpha(X) \end{cases}$$

Value-at-Risk, VaR, can be difficult to optimize for certain distributions. If a normal distribution is assumed for the random variable, \mathbf{X} , then VaR is proportional to the standard deviation. If the assumption is made then $X \sim N(\mu, \sigma^2)$ and the cumulative distribution function of X is $F_X(z)$. Equation 5.3 shows the formula for solving with $k(\alpha) = \sqrt{2} \operatorname{erf}^{-1}(2\alpha - 1)$ and $\operatorname{erf}(z) = (2\sqrt{\pi}) \int_0^z e^{-t^2} dt$. CVaR is then also proportional to the standard deviation, with the formula in Equation 5.4 and $k_\alpha(\alpha) = (\sqrt{2\pi} \exp(\operatorname{erf}^{-1}(2\alpha - 1))^2(1 - \alpha))^{-1}$

$$VaR_\alpha(X) = F_X^{-1}(\alpha) = \mu + k(\alpha)\sigma \quad (5.3)$$

$$CVaR_\alpha(X) = E[X|X \geq VaR_\alpha(X)] = \mu + k_1(\alpha)\sigma \quad (5.4)$$

For the full derivations, see Rockafellar [159]. For more details on closed form solutions for different distributions, like Weibull and gaussian, see [160]. For example, the Weibull distribution exists with a closed form solution for the CVaR, as detailed in [160].

5.4.2 Energy Risk

Consider the battery energy level risk as the user risk function for energy. The risk of the battery running out and the vehicle failing before the mission is complete is a function of the state and actions of the vehicle. The energy used from the battery changes depending on small factors in the trajectory of states and the inputs applied. Energy is a traverse-dependent metric, as detailed in [151]. However, if the running measure of the energy is assumed a Gaussian distribution, then the energy risk can be measured from not only the predicted values, but the updated probability distribution. A local-dependence constraint is assumed in addition to the Gaussian assumption.

Previous work demonstrated the CVaR metric in [63] and [66]. The formulation for the risk function from the latter is considered and modified to fit the energy model of interest. In this formulation, the assumption is that the State-of-Charge, SOC, value is provided by the electric powertrain observer. The SOC is a value between 0 and 100 when viewed as a percentage, and for this work is essentially a linearized estimation of available energy remaining. Modern small aerial systems often operate with a Lithium Polymer battery and the SOC estimate here is as simplified way to estimate the energy available from the battery. Determining the energy available from batteries is still a topic of discussion, for instance see [131], but a general rule of thumb currently is to not go below 20-30% remaining SOC.

The energy risk metric function is defined in Equation 5.5. The risk function G takes the input, x , and the current limit, e , and outputs a numerical value for use in the CVaR formula. Before this is done, though, the additional parameters in the formula must be

Table 5.2: Energy Risk Function Parameters Selected

	γ	b	λ
Value	10	20	1

examined. The function has two parameters, γ , and λ . However, λ is clearly limited to a range of values greater than 0 and is explainable as a bound to the range of $x - e$. The γ remains as a tunable parameter that controls the scaling of the risk quantity in the output.

$$G(x) = \exp\left(\frac{\gamma}{\max\{x - b, \lambda\}}\right) \quad (5.5)$$

The parameters of b , λ , and γ can be solved for the function shape and risk output desired. After initial experimentation, the values are selected as $b = 10$, $\lambda = 10$, and $\gamma \in [0, 30]$ When selecting $b = 20$ and $\lambda = 1$ as initially desired since a SOC at 20% is often considered a limit, the exponential function scales too dramatically at the low ends of x . The function is shown in Figure 5.3 and Figure 5.4 for the set value of γ . The value of γ is set equal to 10, which maxes the risk function out at the value of 10. The formula for determining the value is found in Equation 5.6 and the final parameters are shown in Table 5.2.

$$\gamma = c * \log(R_{max}) \quad (5.6)$$

Considering the case of the variable X we can examine the distribution after the data is passed through the risk function, $G(X)$. First, it is assumed that the estimate of energy available is the percentage of SOC and that it is a random variable that can be observed to match a Gaussian distribution. The example shown in Figure 5.5 assumes a mean of 50 W-hr, and a standard deviation of 2. If this distribution is sampled and passed through the risk function, $G(\mathbf{X})$, the resulting distribution is seen in Figure 5.5. The distribution now represents the distribution of *risk* for the current energy state. The CVaR metric introduced previously is now used to map the distribution of risk to a single scalar value. As a demon-

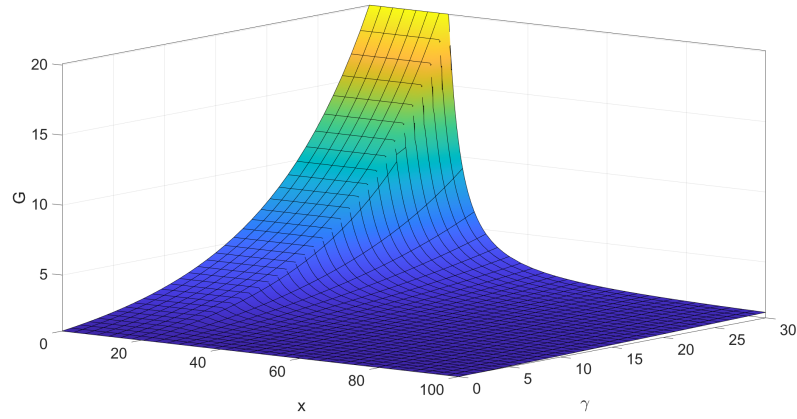


Figure 5.3: Energy Risk Function with varying γ

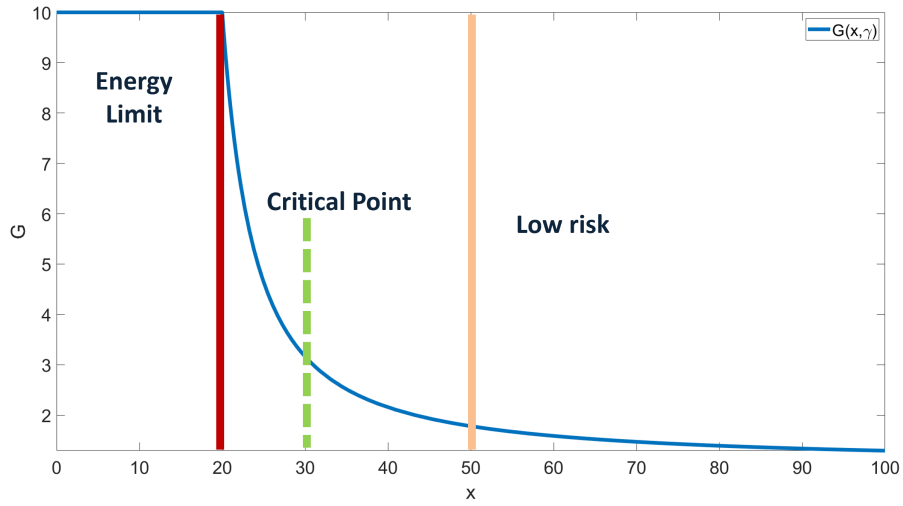


Figure 5.4: Energy Risk Function

stration, the distribution is modified to have a standard deviation of 5, and the risk metric for all value across the distribution is plotted over the distribution seen in Figure 5.6. For this distribution, samples with a risk of 1, which is scaled to 10 in the risk function, are unlikely but possible. This is seen more clearly in the plot of the risk distribution where the mean, VaR, and CVaR can be compared, seen in Figure 5.7. The CVaR critically accounts for the worst-case scenarios, resulting in a value of around 3.2 compared to the mean value of around 2.25.

Consider fitting a distribution to a sampling of data represented in a histogram. Distributions such as Gaussian, gamma, Weibull, and exponential are shown in Figure A.1

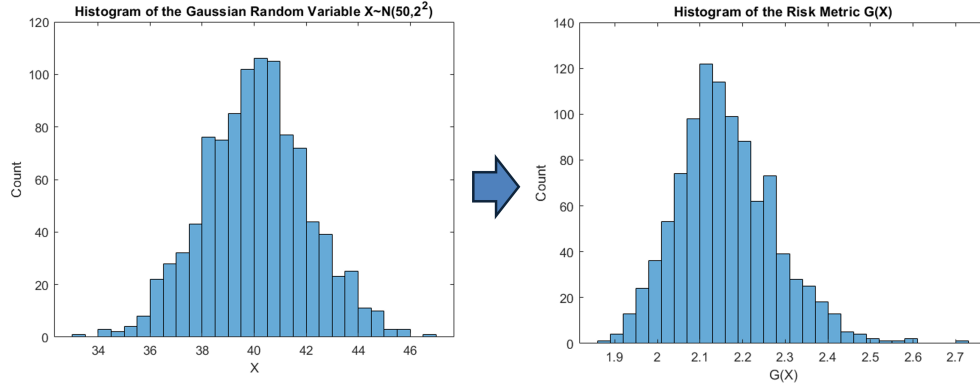


Figure 5.5: Gaussian Random Variable, X and Risk Distribution, $G(X)$

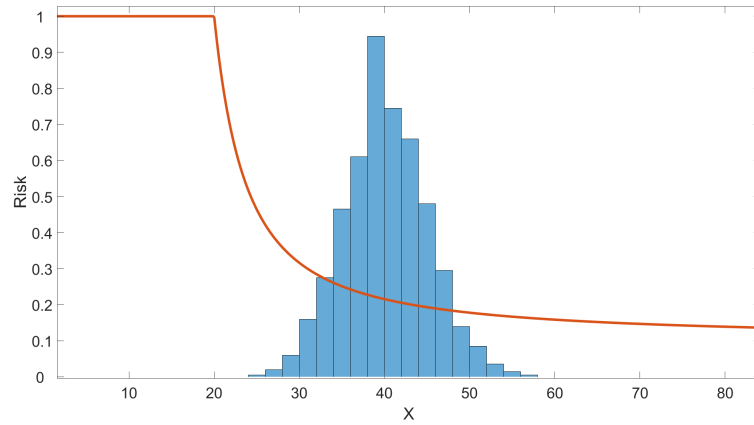


Figure 5.6: Distribution of Energy and Resulting CVaR Risk Value, scaled to a max of 1

using the Python package, Fitter⁵. Two examples of distribution shape are shown in Figure A.1, when the distribution closely resembles a Gaussian and when high-risk values shift the distribution to the right. Alternate distributions are able to catch the right tail values, such as Weibull and gamma. However, when more high-risk samples collect at the value of 10, all the way to the right, no single distribution mode can fit the samples well. Attempts were made to fit the Gaussian and Weibull distributions, and the solutions to the CVaR calculation of those two distributions are available in past sections and in the Appendices. However, the decision is made to use the sampling formula for CVaR for the risk distributions to catch the tail-end risks better. If the X is a sampling of the distribution \mathbf{X} .

⁵<https://fitter.readthedocs.io/en/latest>

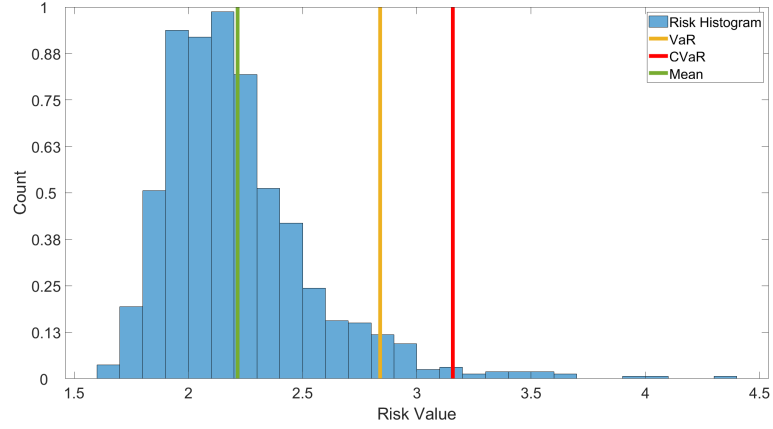


Figure 5.7: Distribution of Risk and Values of CVaR and VaR from input $X \sim \mathcal{N}(40, 5^2)$

The VaR and CVaR can be calculated from the sampled set X through Equation 5.7 and Equation 5.8, which represents the α quantile of the data and the mean of the α quantile.

$$VaR(X, \alpha) = x \mid x \geq \alpha * X \quad (5.7)$$

$$CVaR(X, \alpha) = mean(x \mid x \geq VaR(X, \alpha)) \quad (5.8)$$

5.4.3 Collision Risk

Previous approaches that measured the risk of collision include Ochoa in [65], where a risk proximity metric, $m(x, y, z) = \min(d/d_{thresh}, 1)$ is used for normalizing the distance and mapping to a risk value. However, the uncertainty of obstacle and system positions was not included in the risk metric, m , and the function is linear, which misses the nonlinearity of risk. Therefore, the decision is made to model collision similarly to the energy risk.

The risk of a collision is defined through a distribution of the distance to the closest obstruction, $d \sim \mathcal{N}(\mu, \sigma^2)$. The distribution is mapped by calculating the nearest structure using the nearest neighbors, and then projecting the position to the terrain surface model. Both the structure data and the terrain model output a mean and standard deviation for distance, and the most constraining values are used for forming the distance distribution

Table 5.3: Collision Risk Function Parameters Selected

	a	γ	b	c	ρ
Value	10	1.5	5	0	15

shown in Figure 5.10. The risk function is mirrored from the energy risk formula and defined in Equation 5.9. The parameters used for experiments are seen in Table 5.3 and a plot of the risk function is shown in Figure 5.9. The distance distribution mapped to the risk distribution, and the resulting CVaR values are seen in Figure 5.10.

$$R(d) \longrightarrow G(d) = A * e^{-\gamma \max(d-b, c)/\rho} \quad (5.9)$$

The parameters are selected by examining the open-loop dynamics that were first introduced in Chapter 4. Wind and noise are added to the dynamic maneuvers from the motion primitives. The wind magnitudes are randomly changed from zero to five meters per second in a random direction, with the additional wind gusts formed from the Dryden model. Acceleration noise was randomly selected between zero and 0.25. The resulting position error distribution is seen in Figure 5.8. The distribution has a mean of 24.1 and a standard deviation of around 10. Therefore, the collision risk function is defined such that under windy conditions and open-loop flight the risk will indicate a worst-case risk that is avoided if possible. The collision risk function with the final parameters is seen in Equation 5.9.

5.4.4 Situational Awareness Risk

Previous literature has investigated the sensor coverage problem as a viewpoint orienteering optimization. The method could be using a sensor radius, ray tracing, or an offline viewpoint field where the space is discretized to a finite set of positions and sensor angle. The use of a View Information Field for accurate scene reconstruction is seen in [161]. The actual orientation of the aerial system is solved in planning in [162] and [163]. These

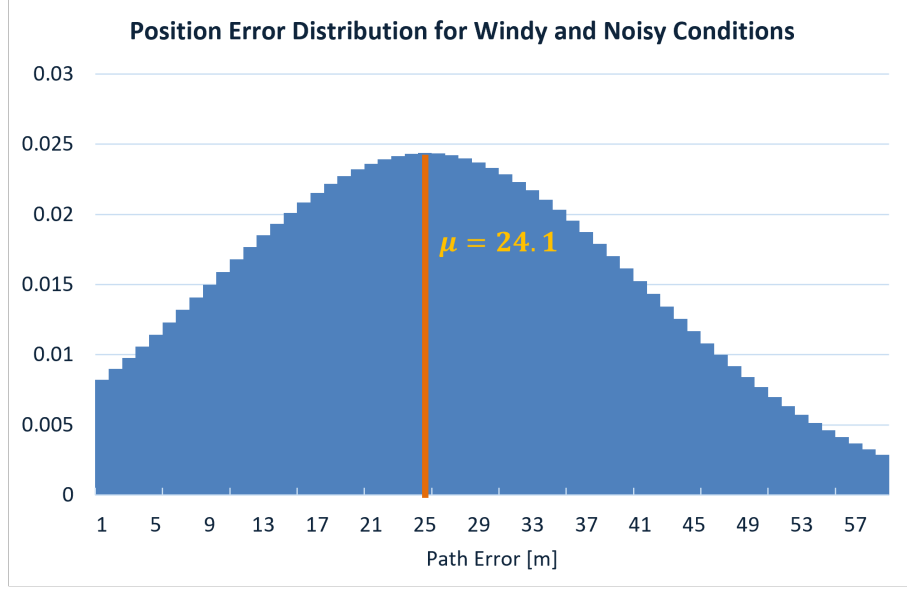


Figure 5.8: Open-Loop Dynamics under Wind and Noise Disturbances

methods are successful but computationally expensive.

A common approach is to model the view of an onboard camera sensor using the scale of coverage through the formula in Equation 5.10 where f is the camera focal length, and H is the flight altitude above ground. The sensor size of the camera and pixel size can then be translated to the coverage on the ground and an estimate of whether objects can be detected in the image.

$$scale = \frac{f}{H} \quad (5.10)$$

However, those constraints limit the trajectory planner's ability to navigate based on the previous two risks. Therefore, the viewpoint risk or situational awareness risk is defined as a spatial and attitude risk or cost that is reduced by controls that maneuver the system close to landmarks. This is unique to the previous two risk functions as the CVaR metric is not used.

The risks are defined by distributions of the distance to the landmark, $d \sim \mathcal{N}(\mu_d, \sigma_d^2)$, and the angle to landmark, $\phi \sim \mathcal{N}(\mu_\phi, \sigma_\phi^2)$. The distributions are passed through the knowledge function in Equation 5.11, and the mean is taken as the likely amount of in-

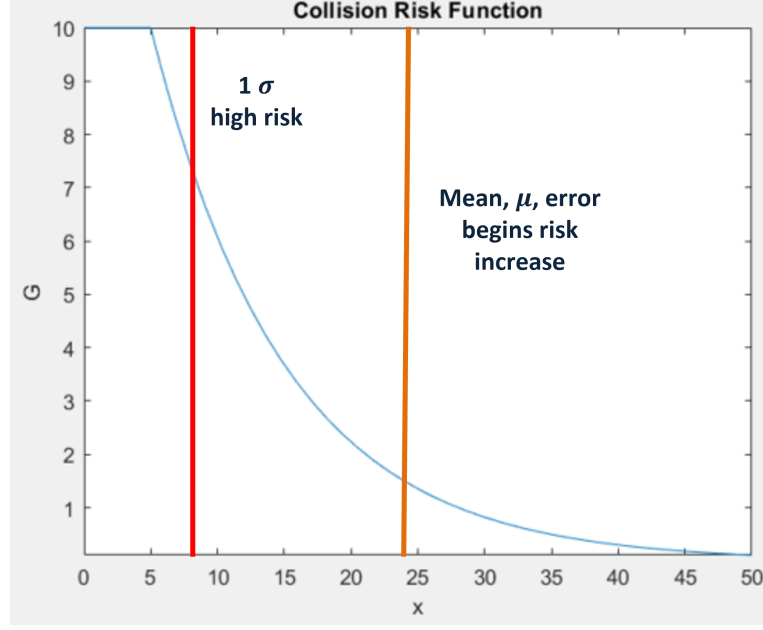


Figure 5.9: Collision Risk Function

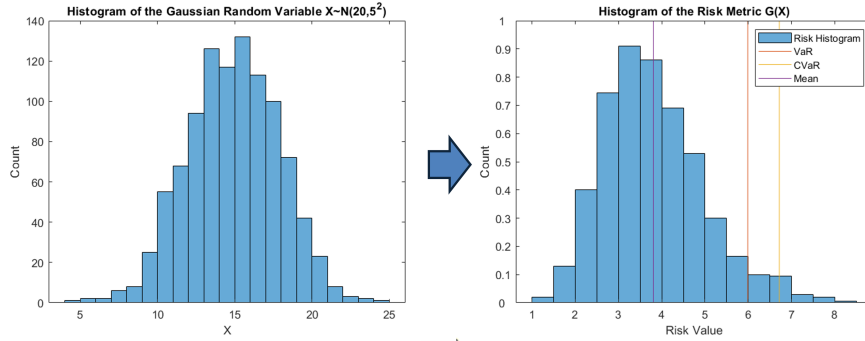


Figure 5.10: Distance Normal Distribution and resulting Risk Distribution and CVaR

formation acquired on the landmark, L_i . Each landmark updates for the remaining risk through the equation, Equation 5.12. The knowledge function and risk function plots for a single landmark, i , are seen respectively in Figure 5.11 and Figure 5.12. The total risk for all landmarks is taken to be the mean in order to balance the total risk with the improvement of reducing the risk of a single landmark.

$$k(d, i) = e^{-f \max(d-t, c)} \quad (5.11)$$

$$r(k, i) = e^{\gamma/\max(k-b, l)} \quad (5.12)$$

Insight from previous work by Kurz et al. in [164] is used to make assumptions on the viewpoint knowledge function. The same camera system is assumed to be accessible as the Canon EOS 1Ds Mark III camera with a 50mm Zeiss lens. The knowledge function parameters are selected based on these assumptions and the limitations assumed for acquiring useful information about a landmark through the camera. The 1000-meter and 150-meter targets are seen in the viewpoint knowledge plot in Figure 5.11. The number of sensor samples is matched with the risk level from the knowledge metric, and the results are seen in the viewpoint risk function in Figure 5.12. The targets of 50% and 90% knowledge are shown to be the critical points for the risk function.

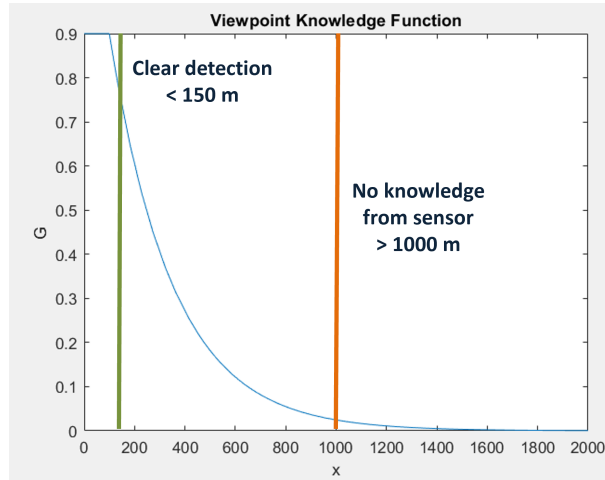


Figure 5.11: Viewpoint Knowledge Risk Reduction Function

5.4.5 Combined Risks

As mentioned earlier, the overall risk of the trajectory represents the probability of mission failure. For this reason, critical targets are mapped to realistic parameters for each of the risk metrics. The energy, collision, and viewpoint risk take internal and external inputs to decide how the worst-case scenarios map to a single scalar value, respectively. The three

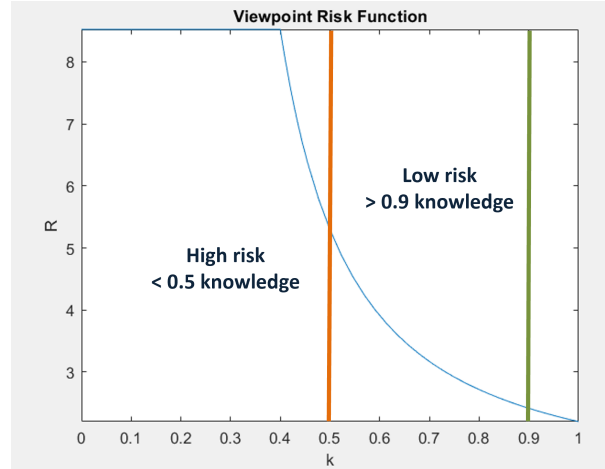


Figure 5.12: Viewpoint Risk Function

can be combined by mapping each metric independently or by combining them together.

A summary of the risk metrics used in this work is found in Figure 5.13. Each metric, as seen in the risk mapping and function columns, has unique parameters which were chosen in the previous sections. Visual examples of the risks are seen with demonstrations of when they cause high-risk values. The energy risk grows for long-range flights relative to the energy available. The collision risk increases in tight spaces where terrain or structures must be flown by to arrive at the goal. The viewpoint risk is only necessary when landmarks are included, but if so, the initial risk is from the lack of viewpoint knowledge.

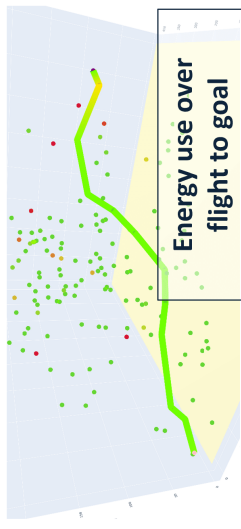
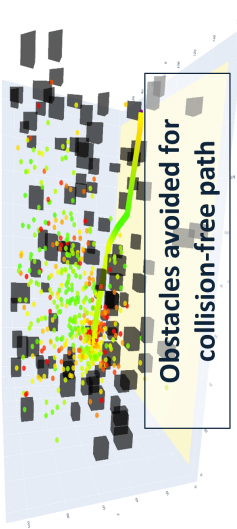
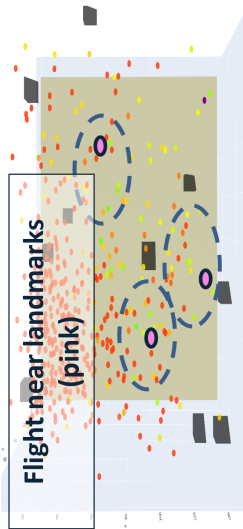
Risk	Risk Mapping	Risk Metric / Function	Risk Value	Flight Demonstration in Risk-aware SST
Energy	$\mathbf{e} \sim N(\mu_e, \sigma_e^2)$	$G(\mathbf{e}) = e^{\left(\frac{\gamma}{\max(\mathbf{e}-b, c)}\right)}$	$r = CVaR(\mathbf{g}, \alpha)$	 <p>Energy use over flight to goal</p>
Collision	$\mathbf{d} \sim N(\mu_d, \sigma_d^2)$	$\mathbf{g} = G(\mathbf{d}) = Ae^{\left(-\gamma \frac{\max(\mathbf{d}-b, c)}{\rho}\right)}$	$r = CVaR(\mathbf{g}, \alpha)$	 <p>Obstacles avoided for collision-free path</p>
Viewpoint / SA	$\mathbf{d}_l \sim N(\mu_l, \sigma_l^2)$	$v = e^{(-f \max(\mathbf{d}, -t, c))}$	$r = e^{\left(\frac{\gamma}{\max(k-b, l)}\right)}$	 <p>Flight near landmarks (pink)</p>

Figure 5.13: Risks Summary

5.5 Risk-Aware Planning

The primary objective is safer flight, through the SAFER framework. Trajectory planning literature has slowly drifted towards the use of risk-aware planning to approach safe flight in general, whether for the safety of the vehicle or the surrounding people and property. Jasour in [165] defined risk-aware planning as the task of finding "state trajectories and control policy to satisfy safety constraints and control objectives, in the presence of probabilistic uncertainties." Xiao et al. in [151] generally define *risk* for motion planning as "the probability of the robot not being able to finish the path." Xiao sought to address the formal definition by proposing three major risk categories; locale-, action-, and traverse-dependent risk. This definition and a complete risk-aware planner are introduced and applied to the borehole entry problem, where the three categories are expressed in a "Universe of all risk elements" and labeled for the dependence on risk elements or agent state history. Sharma et al. in [166] applied a risk-theoretical measure CVaR to define risk to find safer paths for ground vehicles in disasters. Semantic segmentation is applied to aerial images to represent the map's accessibility, however, this is noted as a risk-neutral approach. Therefore, additional information on the uncertainty of the semantic predictions to form a risk-aware approach. Risk metrics in past works have often used this measure of Conditional-Value-at-Risk (CVaR), which has been used in the past for expected shortfall in the financial risk of investment portfolios. CVAR is tunable using the parameter α , which defines the expected tail loss or likelihood of event failures. Other work has represented risk using the metric of entropy as in [167]. The definition of entropy is modified from the uncertainty degree of a system from information theory to a combination of expectation and entropy.

If the risk can be statistically represented, it removes the arbitrary nature of riskmaps. This is reminiscent of [168] by Gelman and Yao that notes uniform priors or subjective priors are poor and incoherent for Bayesian inference. In many cases, the solution has been to consider a frequentist view of statistics or otherwise assume the powerful Gaussian distri-

bution as a prior. However, there is literature expressing the importance and proper process of leveraging accurate prior information, such as Gelman et al. in [169]. Alternatively, the likelihood of risk of an event can be represented by a mapping developed by NASA, which reduced a decision to three color-coded risk levels as detailed in [170]. If assumptions are required, such as in the case of space robot operations, this systematic mapping may be more productive than attempting to form accurate priors.

The representation of risk requires a formal definition using probability theory and propositional logic. The probability of mission success is then the multiplication of the probability of each successive state conditioned the previous states as in Equation 5.13. The probability of not finishing the path is as in Equation 5.14.

$$P(F) = P(F_f \dots F_0) = \prod_{i=0}^n P(F_i | F_0, \dots, F_{i-1}) \quad (5.13)$$

$$P(\bar{F}) = 1 = \prod_{i=0}^n \prod_{k=1}^r (1 - r_k(s_1, \dots, s_i)) \quad (5.14)$$

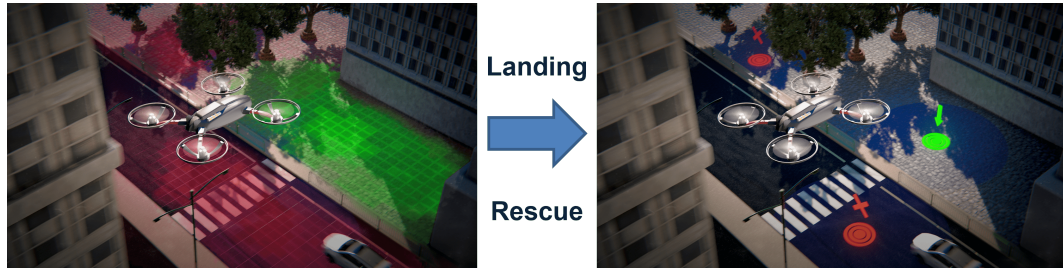


Figure 5.14: Costmap for the Risk of Landing when Risk-aware Planning is applied to a Rescue Mission. Graphics reproduced with the approval of the creator, Alexander Parmley, from <http://www.aparmley.com>

This work features the three risk metrics outlined in the previous section and summarized in Figure 5.13. The overall objective of the risk-aware approach is to minimize the $P(failure)$, which can be considered as in Figure 5.14, through the three independent risk metrics. Each metric maps to a single scalar risk value; however, the risk of a trajectory must be further defined. The three metrics make the problem a multi-objective optimiza-

tion, and the shift of the risk distribution over the trajectory can be combined through risk thresholds, summing the risks, taking the max or mean, or another way.

$$R(\sigma) = \max(r_i) \quad \forall i \in \sigma \quad (5.15)$$

$$R(n) = \max(R_1, R_2, R_3) \quad (5.16)$$

5.5.1 Background

Planning within a riskmap or costmap is not trivial, particularly in the applications of interest in this work. Planning algorithms for the configuration of the space of an aerial system can be computationally expensive, depending on the assumptions and constraints imposed. Planning methods include heuristic grid search approaches like Dijkstra and A*, stochastic graph search like Markov Decision Process or RRT, geometry optimized algorithms like Model Predictive Control, or approximated dynamics like Differential Dynamic Programming.

Previous works have applied advanced techniques such as memory-less planners, graph neural networks, or factor graphs. In addition, alternative routing approaches, such as by Scherer et al. in [34], considered dynamic events during flight and the uncertainty in path predictions with unknown vehicle dynamics. Based on the success of the SST algorithm in Chapter 4, the SST algorithm is selected to integrate into the risk-aware planning framework. Three planning algorithms, EASST, iEASST, iRASST, are introduced and explored using an object-oriented software environment. Additions to the planner are made until all risk metrics are included and the risk-aware planner is capable of efficient planning.

5.5.2 Energy-aware SST

The first planner leverages the energy risk metric introduced in the previous section. Energy is tracked along the tree expansion using a Kalman Filter, which is briefly described in

Equation 5.17 as the combination of the measured value, z_k , and the previous estimation, \hat{X}_{k-1} , through the linear Kalman gain, K . The predicted energy updates are estimated by factoring the mean and standard deviation of energy in hover with the time and speed of the dynamic states. The Kalman filter helps to smooth the estimates and tracks the Gaussian uncertainty, especially over long flight prediction steps.

$$\hat{x}_k = K * Z_k + (1 - K) * \hat{X}_{k-1} \quad (5.17)$$

The Kalman filter requires an estimate for the estimate propagation over time. For the energy use, a scaled hover energy is used by scaling by the thrust and angle setpoints of the motion primitive and using offline data to form a mean, μ_e , and standard deviation, σ_e for the estimate. An example of the energy and standard deviation over time is shown in Figure 5.15.

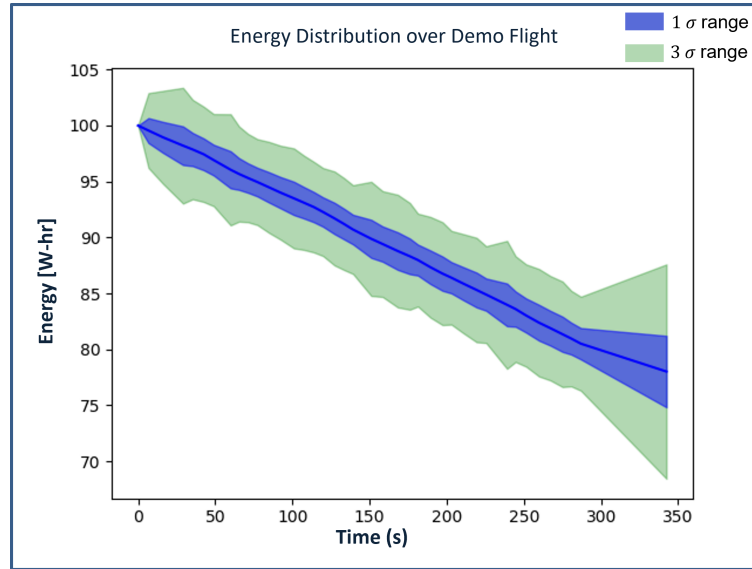


Figure 5.15: Mean Energy and one and three σ for Flight through Generated Map

The algorithm will stop the search at a node that tracks the mean SOC to be below 20% or if the risk is equal to or greater than the limit 10. Figure 5.16 demonstrates the flight of the SST and Energy-aware SST algorithms by simulating the quadcopter model through

the optimal paths found after 500 iterations. The total energy use for the SST algorithm approaches 33%, while the EASST algorithm is only 20%. This is achieved by finding paths that minimize the risk of energy, as opposed to the euclidean distance. However, EASST lacks insight into when and where to expand nodes, causing paths that get stuck in local minimum during finite-time runs with default parameters. Therefore, improvements are desired that can improve the speed of convergence and solution quality.

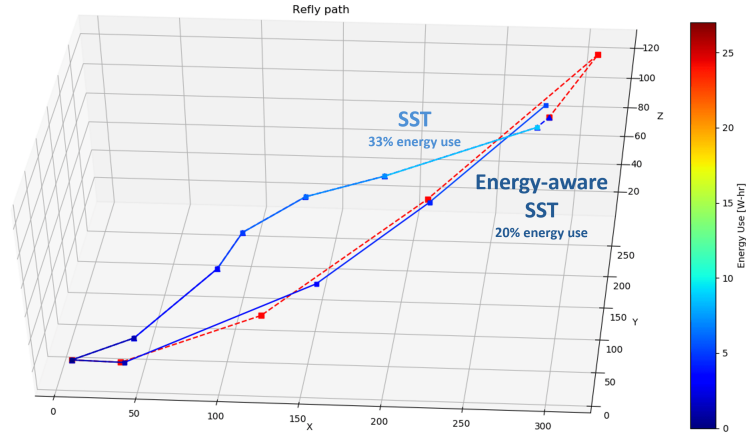


Figure 5.16: Reflying Paths and Tracked Energy for SST and EASST Algorithms

5.5.3 Informed Energy-Aware SST

Arslan and Tsiotras in [171] leveraged machine learning-guided predictions for efficient sampling and steering. In [172], a GAN and LSTM are used to generate efficient heuristics in a 2D sampling-based planner to improve the sampling process towards the goal. Work by Wang et al. in [173] demonstrated the NRRT* that utilizes a nonuniform sampling distribution generated from a CNN model. Another technique for learning leveraged the latent space of planning problems that can be learned as in [125] where "the learned latent space is constructed through an auto encoding network, a dynamics network, and a collision checking network, which mirrors the three main algorithmic primitives of SBMP, namely state sampling, local steering, and collision checking" [125].

Informed sampling in [174] demonstrates an improvement to the random sampling of

stochastic search algorithms through a minimum-path-length problem using the L2 informed set from the L2 norm. Informed sampling approaches were generally defined [111] as methods that reject samples that do not improve the current best estimate of the solution. Informed RRT* in [175] uses direct informed sampling to speed up the time to find the asymptotically optimal path from RRT*. The difference exists when the first path is found, and the sampling focuses on the areas where the path can be improved. This is done through direct sampling of the ellipsoidal heuristic, X_f . Improved convergence rates can be achieved by using informed trees as in BIT* and RABIT* [120].

Littlefield and Bekris in [176] shows how the SST can be improved with informed sampling demonstrated on dynamic systems in the real world. Littlefield and Bekris in [177] show how Dominance-Informed Regions can provide informative sampling that improves convergence for kinodynamic motion planning. Then in [178], Sivaramakrishnan and Littlefield develop an algorithm to appropriately balance an exploitation-exploration trade-off using neural networks to infer local maneuvers for a robotic system with dynamics. However, the informed SST algorithm detailed by Littlefield is selected as a balance of algorithm complexity and improvement.

Informed SST introduces five key advancements that are suited well for field robots as detailed in [176]. One, a deterministic sampling is used to decide which node to select for forward propagation of the dynamic model. Two, heuristics are defined to guide the exploration through both a running cost and a cost-to-goal, or heuristic cost, comparison. Three, the motion primitive maneuvers are mixed with random control selections to keep probabilistic completeness guarantees. Four, the locally optimal control decision is selected from a set of forward propagation maneuvers that are compared to local witness nodes by risk value and heuristic value. Fifth, a branch and bound technique forces the search to go towards low cost-to-go by continuously sampling along paths with low heuristics

As demonstrated by Littlefield in [176, 128], a heuristic can improve convergence. Just as for the A* algorithm, the heuristic must be admissible and consistent. The main idea is

that the heuristic function, $h(n)$, is defined to be a lower bound estimate of the optimum cost to the goal as seen in Equation 5.18. The heuristic function influences which nodes are selected by sorting all nodes in two priority queues. The priority queues use a heap structure to efficiently sort nodes with the lowest combined cost at the top of the list. The nodes with the lowest cost, $f(n)$, are continuously selected and kept at the front of the list in the node expansion. The *MonteCarloProp* function leverages the name even further by forward propagating from the node multiple times using either the motion primitives or random controls. The number of samples, m , and the likelihood the sample is chosen for further expansion depends on the quality function, detailed in Equation 5.19. The node's variable, p , is the priority value which increases from 1 to 10 based on the number of times it has been selected. The node's cost, or risk value, is indicated by the variable c .

$$h(n) \leq h^*(n) \quad (5.18)$$

$$q(n) = \left(\frac{1}{n.c/goal.c + h(n)/h(n_0)} \right)^{n.p} \quad (5.19)$$

The heuristic function must take into consideration one or more of the risk metrics introduced earlier, and therefore three components of the heuristic function are introduced. The energy to the goal heuristic defined in Equation 5.21 is the energy distribution predicted to travel from the current node to the goal node. The assumption is that a straight line to the goal is flown at the current average speed so that the scaled hover energy rate can be summed over the full time of flight. The collision heuristic is a function of previous distances to obstacles, while the viewpoint heuristic is calculated by measuring the difference in viewpoint knowledge of the closest landmark.

$$h(n) = h(n)_e + h(n)_c + h(n)_v \quad (5.20)$$

$$h(n)_e = CVaR(G(\mathbf{E})) \mid \mathbf{E} \longrightarrow d(n, g) \quad (5.21)$$

The total path risk is measured in two ways. For one, the overall path risk can be defined as the risk at the final node, the goal node. Alternatively, the risk can be defined as the risk at the β quantile of the risk set or the mean of the risk set.

$$R(\sigma) = R(n_{goal}) \quad (5.22)$$

$$R(\sigma) = R(quantile(\sigma, \beta)) \text{ or } R(mean(\sigma)) \quad (5.23)$$

First, the informed heuristics for the energy risk metric is explored and compared to the previous algorithms introduced. The algorithm named iEASST shows major improvements for the EASST algorithm and is favored over the original SST for most maps of interest.

Generated Maps

The iEASST, EASST, and SST algorithms are compared for performance and consistency over randomly generated maps and urban maps. First, the algorithms are evaluated over four generated maps. Map "14" is the "hole-in-the-wall" map introduced in Chapter 4. The maps named with the format "r#-##" are randomly generated maps that are defined by the scale, from one to 5 times in size, and the number of obstacles, from zero to 200 obstacles. The scale increases the bounds and the distance of the goal from the start. The obstacles are randomly dispersed in the environment and range from 5 to 25 meters in height and width. The results for the planners on all four maps are shown in Figure 5.17 and detail the best performers in the time to find a solution, the number of iterations the first solution was found, and the total energy used across the trajectory. Detailed outputs are found in Table B.2.

The iEASST algorithm is shown to produce lower energy trajectories consistently across

the maps while requiring fewer iterations. In addition, while the EASST algorithm has trouble finding solutions in the set number of iterations, the iEASST fixes the problems and outperforms the SST algorithm, as seen in Table 5.4. The reason for this has to do with how the energy risk metric changes the expansion of the tree and reduces the sparsity across the entire map. Sparse nodes exist along energy efficient directions, but iEASST is needed to speed up the search with the energy risk metrics to consistently find solutions. Figure 5.17 details the performance in detail across the four generated maps, where the iEASST algorithm is consistently better than EASST and SST for total energy use. EASST algorithms are expected to find similar trajectories over infinite time runs, but in finite-time performance iEASST outperforms by EASST. However, the settings for the experiments and the costly dynamic functions for the randomized control states result in iEASST finding the first solution in more time than the other algorithms. Therefore, reducing the number of iterations or setting tighter time constraints could disrupt the iEASST algorithm's performance. Therefore, future work could investigate the optimized parameters and performance limitations for each map.

Table 5.4: SST, iEASST, and EASST Success Rate on 3D generated maps

<i>60 runs Each</i>	SST	iEASST	EASST
All [%]	95.56	98.89	66.67

RUM Maps

The three algorithms are evaluated across the three urban map models generated in Chapter 3. iEASST performs the best, as shown by the metrics in Table 5.5 and Table 5.6. In particular, the performance on the Atlanta maps demonstrates the exceptional ability to find solutions over large maps, where the sparsity of the original SST algorithm is lacking until it is run for more iterations. The iEASST algorithm's use of the heuristics leads to a sparse nature of nodes around the goal, as opposed to over the entire search space.

Results in Figure 5.17 show how in generated 3D maps, the iEASST algorithm may

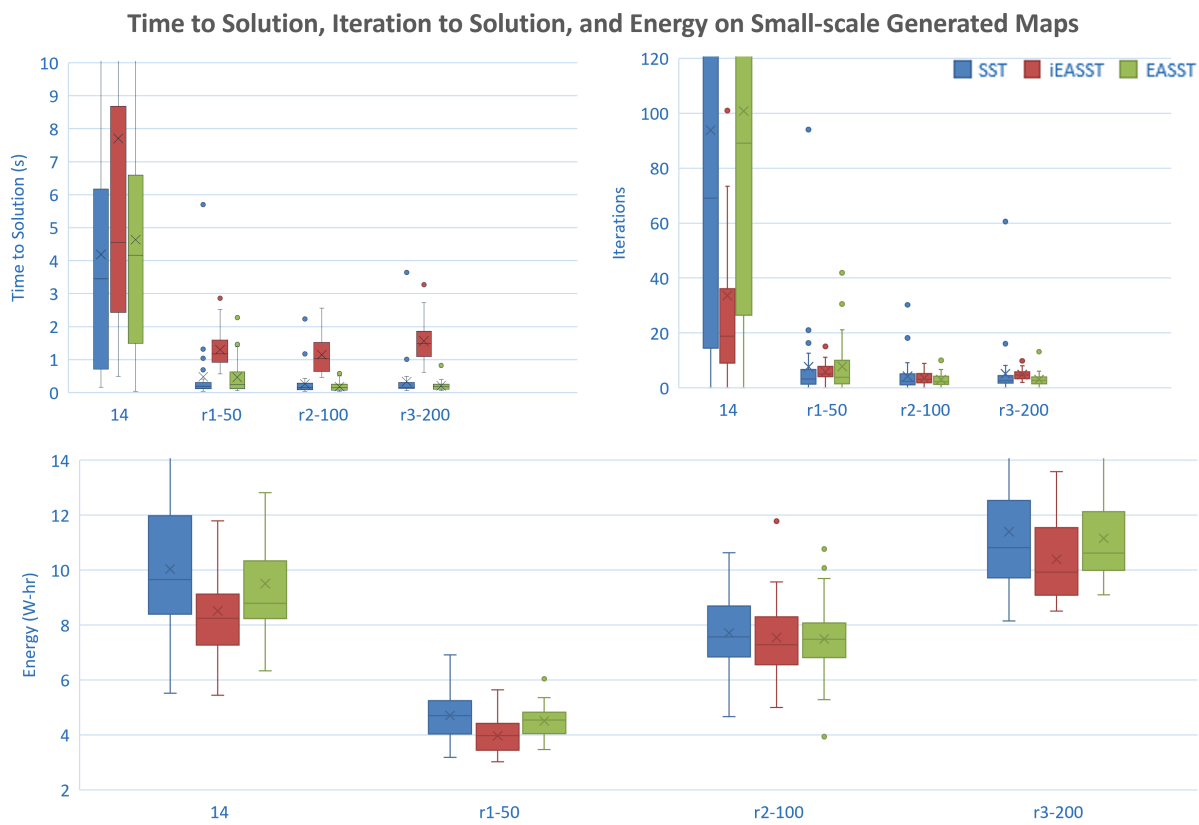


Figure 5.17: SST, iEASST, and EASST performance on 3D generated maps

take longer in some maps to find the first solution. However, the final solution within the finite-time experiments is much improved regarding energy use over the flight. Figure 5.18 examines the performance over the three rapid urban map models. The results in Figure 5.18 similarly indicate the success of the iEASST algorithm. The iEASST algorithm finds trajectories in the three urban maps, Atlanta, Los Angeles, and New York City, that are all less energy and time to fly. The energy of the flight is 30% improved over the EASST algorithm, and the flight path is 15% improved. The only map in which the iEASST algorithm does not provide an average improvement in these metrics is the NYC map. The NYC map has many obstacles in the environment, and the planner must balance the constraints of avoiding collisions with the risk of energy.

The performance on the Atlanta map in Figure 5.19 shows additional insight into the performance of iEASST. First, it is clear that the SST algorithm is now lacking behind the other algorithms when it comes to the trajectories' flight time and energy. When wind and noise conditions increase, the range of solution quality grows and ultimately begins to fail. The EASST algorithm experiences similar difficulties in finding a solution within the 3000 iterations, but the solutions found perform significantly better than SST. Most importantly, however, is the performance of iEASST. The algorithm consistently performs well under windy and noisy conditions, finding the closest to an optimal path within 100 seconds of flight and 20 W-hr of energy. Wind conditions over four m/s, with gusts approaching six m/s, and internal noise with a standard deviation of over 0.4 limit the planner's ability but do not constrain it. In fact, while the flight time is generally increased, the energy of the flight stays the same. Therefore, the planner can, through local heuristics and witness nodes that experience the effect of wind and noise, find alternative paths that minimize the energy risk.

The performance of the iEASST algorithm is improved as compared to the EASST algorithm. This is achieved by adding the informed heuristics and the improved branch and bound functions. However, the algorithm optimizes with respect to a single risk factor,

Table 5.5: SST, iEASST, and EASST performance on urban maps

	SST	iEASST	EASST
ATL [%]	75	92	25
LA [%]	48	41	33

Table 5.6: SST, iEASST, and EASST Energy and Flight Time in the Atlanta urban map

<i>Map: ATL</i>	SST	iEASST	EASST
Avg Energy (W-hr)	89.53	52.41	66.88
Avg Flight Time (s)	628.45	367.29	432.36
Success (%)	82%	100%	27

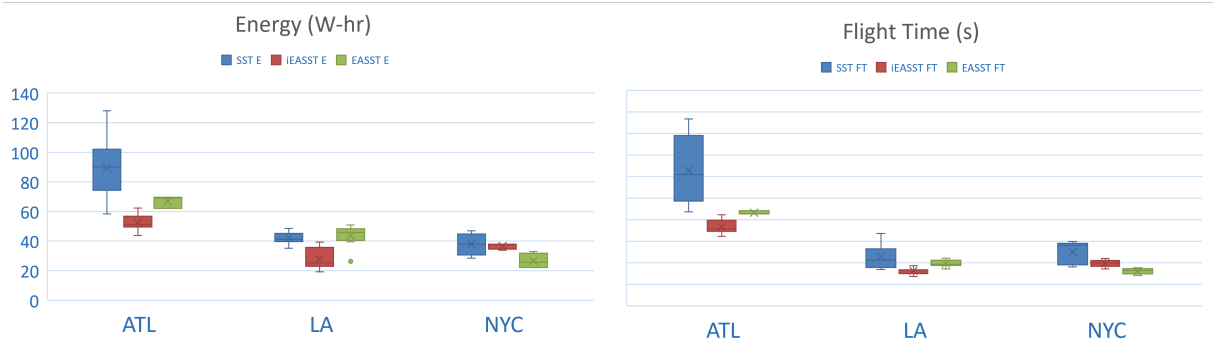


Figure 5.18: SST, iEASST, EASST Energy and Flight Time in urban maps

energy, which is not always the best decision, for example see the results in the NYC map. So far, the risk of collision and data collection were not included. The following section introduces the last version of the planner, which considers all risk metrics during optimal trajectory planning.

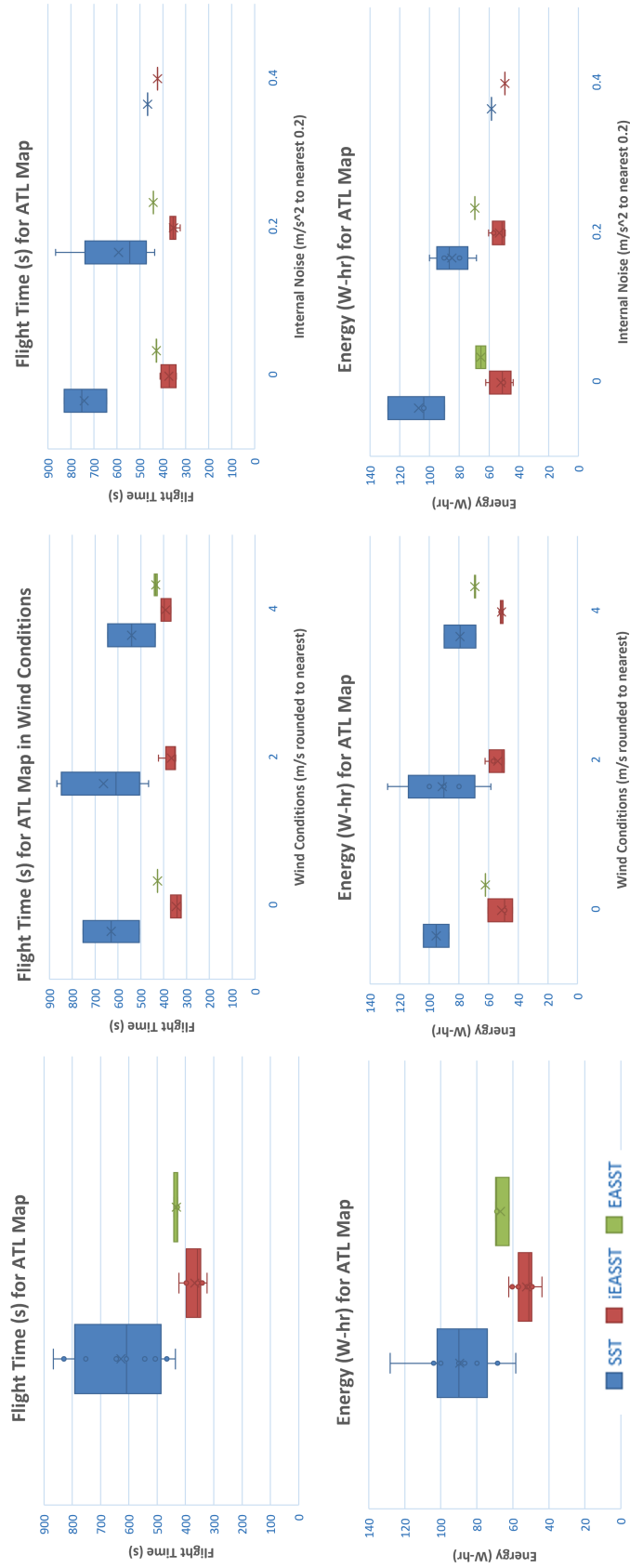


Figure 5.19: SST, iEASST, and EASST performance in the Atlanta urban map

5.5.4 Informed Multi-Risk-Aware SST for Aerial Disaster Response

The final algorithm, which combines all the risk metrics and planning algorithm improvements together, is iRASST. The algorithm is detailed in algorithm 6, where the informed SST additions have been added such as *BestHeuristic* and *BranchAndBound*, and the core *multi-risk* additions have been added within the *CalculateRisk* and *CalculateHeuristic* functions, seen in algorithm 5 and algorithm 4 respectively. The algorithms provide the general structure but are condensed in this work. More details into the best data structures and efficient implementation methods are found in [128] and [176]. All the risk metrics are now represented as soft constraints, which forces the algorithm to find trajectories that balance the three risks for an optimal path. The heuristic function shown in algorithm 4 remains a risk estimate of only energy, as to keep the admissible and consistent assumptions. The assumption then holds that this is a lower bound of risk, where all viewpoints are achieved and no collisions occur while flying straight line, energy-efficient, paths to the goal.

Algorithm 4: *CalculateHeuristic(N)*

```
for lmk in landmarks do
     $d \leftarrow \text{distance}(N.x, lmk) ;$ 
     $t \leftarrow \text{time}(d, V) ;$ 
     $e \leftarrow \text{energy}(t, V) + e ;$ 
end
 $g \leftarrow \text{EnergyRisk}(e_\mu, e_{var})$   $h, r \leftarrow \text{CVaR}(g_\mu, g_{var}) ;$ 
```

Algorithm 5: *CalculateRisk(N)*

```
 $O_\mu, O_{\sigma^2} \leftarrow \text{NearestObstacle}(N) ;$ 
 $c \leftarrow \text{CollisionRisk}(O_\mu, O_{\sigma^2}) ;$ 
 $d_i, \phi_i \leftarrow \text{LandmarkRay}(N) ;$ 
 $v_i \leftarrow \text{ViewpointRisk}(d_i, \phi_i) \quad \forall i \in \text{Landmarks} ;$ 
 $v \leftarrow \text{mean}(v_1, \dots, v_i, \dots, v_k) ;$ 
 $E_\mu, E_{\sigma^2} \leftarrow \text{EnergyUse}(N)$   $e \leftarrow \text{EnergyRisk}(E_\mu, E_{\sigma^2})$ 
 $R \leftarrow \text{RiskCombination}(c, v, e) ;$ 
```

Algorithm 6: iRASST($X, U, x_0, T_{prop}, N, \Delta_W$)

Data: $x_0 \in X, N > 0, \Delta_W$
Result: $p^* = \{t_1, x_1 | \dots | t_f, x_f\} \in \mathcal{P}$
 $risk(p^*) \leq risk(p_i) \forall p_i \in \mathcal{P}$;
 $V_{active} \leftarrow x_0, V_{inactive} \leftarrow \{\}$ $E \leftarrow \{\}$, $G = \{V_{active} \cup V_{inactive}, E\}$;
 $w_0.x \leftarrow x_0, w_0.p = x_0, W \leftarrow \{w_0\}$;
 $P \leftarrow PriorityQueue(\emptyset)$, $Q \leftarrow PriorityQueue(\emptyset)$;
for N **do**
 $x_{selected} \leftarrow InformedBestSelection(X, V_{active}, \Delta_W$;
 $x_{news} \leftarrow MultiMonteCarloProp(x_{selected}, U_{i,\dots,m} T_{prop})$;
 $R \leftarrow CalculateRisk(X_{selected}, \alpha)$;
 $H \leftarrow CalculateHeuristic(X_{selected})$;
 $x_{new} = BestHeuristic(x_{news}, H)$;
 if $CollisionFree(x_{selected}, x_{new})$ **then**
 if $LocallyBestNode(x_{new}, W, \Delta_W, R, H)$ **then**
 if $BranchAndBound(x_{new}, R, H)$ **then**
 $V_{active} \leftarrow V_{active} \cup \{x_{new}\}$;
 $E \leftarrow E \cup \{x_{selected}, x_{new}\}$;
 $PruneDominatedNodes(x_{new}, V, E)$;
 end
 end
 end
 $P, Q \leftarrow UpdatePriorityQueues(x_{new})$
end

All risk metrics generally increase to the max risk in the three regions indicated in Figure 5.20. On the left, red nodes indicate a high viewpoint risk before any detections are made. In the middle, the collision risks increase dramatically as the flight requires maneuvering through the buildings. On the right, flights that continue to search the environment ends at the maximum energy risk.

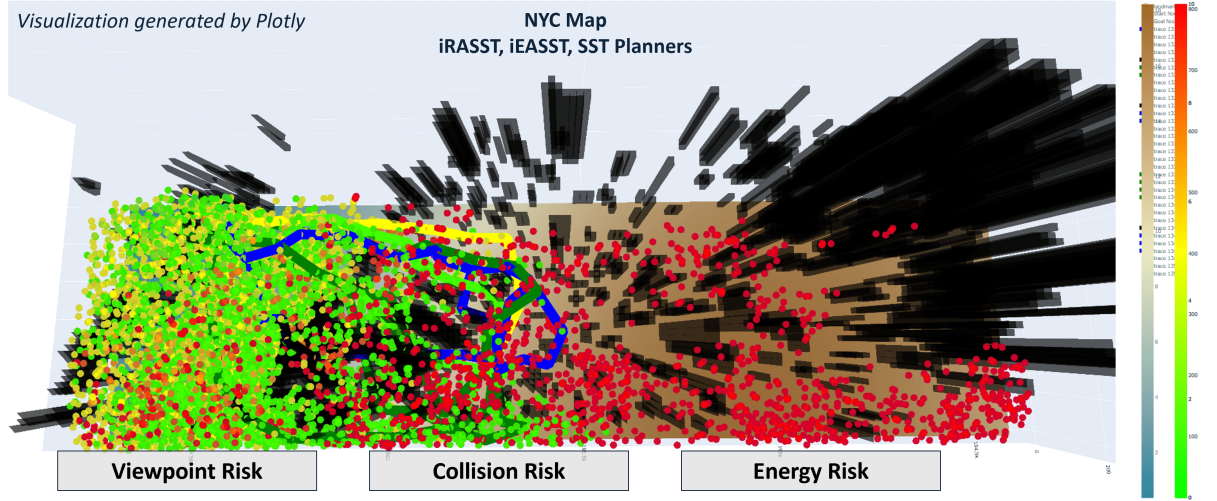


Figure 5.20: iEASST, and iRASST, and SST in New York City demonstrating all risk metrics

The overall results in Table 5.7 show that the algorithm has longer flight times and a higher energy cost when energy risk is not the primary constraint but results in a much lower path risk. The collision and viewpoint risk metrics take priority for these scenarios, meaning that safer flights that ensure landmarks are detected and avoid getting close to structures and terrain perform better on average. For example, in the results from the NYC map the flight path requires 6% more energy and 10% more flight time. However, visualization of the path shows that the extra flight time and energy comes from routes around tight corridors in the NYC map. Easier detection of landmarks may have also played a part in the path selection, but the primary reason is to limit high risk values from potential obstacle collisions. Further investigation using real vehicles or high-fidelity simulation environments is needed to fully understand the tradeoff between risk metrics.

Table 5.7: Path Performance Results for iRASST

	LA			NYC		
	SST	iEASST	iRASST	SST	iEASST	iRASST
Path Energy [W-hr]	37.0	29.39	49.3	35.9	27.9	31.1
Path Flight Time [s]	406	276	293	296	234	243
Path Risk	6.5	6.0	2.8	7.4	6.4	5.0

Table 5.8: Planning Algorithms Playbook

	Long Distances	Many Obstacles	Windy / Gusty
Complexity	SST	SST	EASST
Performance	iEASST	iEASST	iEASST
Consistency	iEASST	iRASST	iRASST
Time	SST	EASST	EASST

5.6 Summary and Future Work

If there are no landmarks nor tall structures, or the viewpoint and collision risk are less important to a stakeholder using the framework, then the iRASST algorithm is not the correct choice. This chapter’s results outline the best circumstances for each planner, and this is summarized in an algorithm *playbook* for deciding which algorithm to use depending on generalized maps in Table 5.8 and specific maps in Table 5.9.

Additional risk metrics could provide more complete guarantees of flight safety. Some

Table 5.9: Planner Playbook on Urban Maps

RUM	Map Details	Planner Outcomes
NYC	Many Obstructions Collision and Energy Risk key	iRASST
LA	Mountainous Terrain Energy and Collision Risk Key	iEASST
ATL	Large Scale with Many Landmarks Energy and Viewpoint Risk Key	iRASST
HIW	Small Scale with Hard to Find Path Collision Risk Key	Informed SST

metrics have been demonstrated in literature in the past, such as Scherer’s use of a time to collision metric in [35] Also, GPS localization using a Geometric Dilution of Precision model was demonstrated as a flight metric by Ochoa in [65]. In addition, consistent communication with other aerial systems or with a base station is critical to relaying updated flight paths or data collected. Engineers involved in this field have suggested that communication should be included when evaluating flight risk and, therefore, should be included in future work.

The Informed SST improvements could also be furthered using tricks from DIRT [128] and TIE [63], or by leveraging sensor entropy reduction [179]. Camera viewpoint models can be improved with the vast literature on the subject. Work by Xiao [62] and Choi [83] have demonstrated camera viewpoint models, while work such as [162] and [163] have solved the viewpoint orienteering problem explicitly.

Recent work by Larsson et al. in [180] has explicitly explored the multi-objective risk optimization for trajectory planning using hierarchical tree structures, signal encoders, and information-theoretic methods. Pareto efficient solutions are presented in the information plane, resulting in a linear optimization problem. Xiao in [62] also investigated multi-objective risk through a Pareto-optimal trajectory search. This can be considered as a linear combination of risk metrics to optimize for mission success over all possible scenarios.

$$R(X) = \alpha R_1(X) + \beta R_2(X) + \chi R_3(X) + \dots$$

One question still remains. How does this perform over realistic disaster response missions? Chapter 6 demonstrates the framework in a realistic scenario to see how well the framework can be leveraged.

DISASTER SITUATIONAL AWARENESS FROM AERIAL TRAJECTORY PLANNING

6.1 Introduction

The risk-aware planner can manage multiple risk metrics to find the most viable solutions to maximize mission success, as shown in Chapter 5. This was demonstrated within the framework that combines rapid urban mapping and safe 3D trajectory planning. The framework now forms a new benchmark to evaluate additional planning algorithms and new risk metrics or uncertainty models. Furthermore, visualization packages allow the 3D maps, trajectory paths, and color-coded risks to be presented quickly. All this is done using open-source software, tools, and publicly available datasets while running on a single laptop. Therefore, it can be claimed that the framework is a novel approach to benchmarking aerial trajectory planning and urban mapping for disaster response applications.

Now that algorithms have been selected and optimized for urban mapping and trajectory planning, the question remains on how the full framework can be leveraged for actual disaster response scenarios. What limitations exist, and how can they be solved for risk-aware planning with limited data when applied to disaster response scenarios? Disaster response scenarios with uncertainty models and optimized planners provide insight into the capabilities of offline planning for response times and success rates. This can feed down into system-level metrics for energy and viewpoint design and performance targets.

In order to answer the overarching research question of whether these algorithms and data selections improve UAS operations for situational awareness, the full framework is implemented using a combination of real data and Monte Carlo simulation.

6.1.1 Disaster Response Areas of Interest

Industry interest is growing for responding to and predicting disasters as shown by companies like Nearmap¹. Recent research has attempted to give insight into likely or already flooded areas using tools FloodNET [95]. However, Geodatahubs used as resources in Chapter 3 often have resources precomputed regarding likely disaster zones. For example, the Georgia Emergency Management & Homeland Security Agency provides geospatial datasets on flood zones in the Atlanta area. The data and environment used for the experiments are shown in Figure 6.1. The raw dataset is a set of polygons representing documented flood zones. The centroid is used to reduce the polygon to a single 3D point to reduce the computations during viewpoint knowledge checks. Just as in Chapter 5, the landmark risks are initialized to be near max risk to influence paths that find the landmarks in the environment.

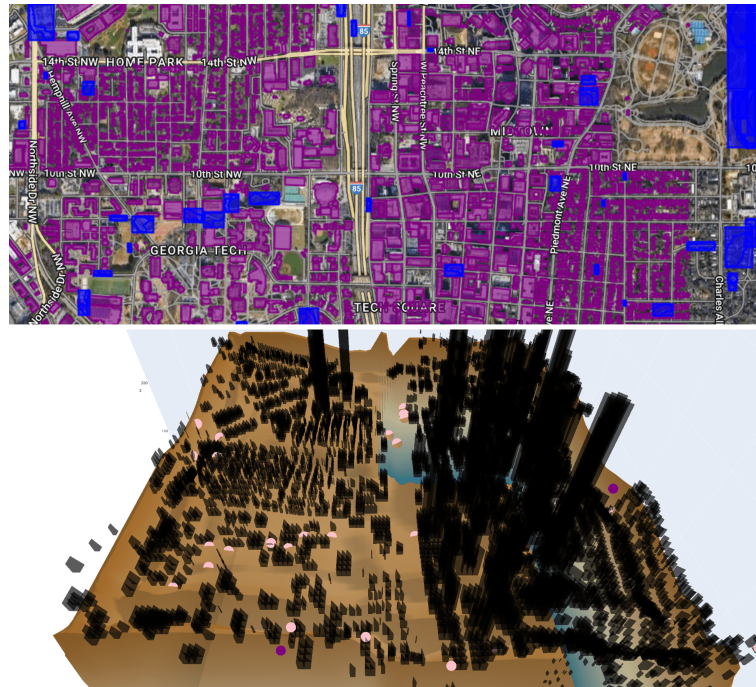


Figure 6.1: Final Demo in Atlanta with Flood Zones in Blue and Structures in Purple, and the mapped landmark points as pink circles

¹<https://www.nearmap.com/us/en/aerial-view-maps-about-us>

Initial results reveal that landmark detection can falter when the landmarks are spread too far from the start and goal location. The energy and collision risks influence the trajectory more quickly, and some landmarks are ignored within the finite-time runs of around 5000 iterations. Therefore, the algorithm needs to have multi-goal characteristics to perform a more realistic trajectory. The objective is not necessarily a point-to-point search anymore, but rather a multi-goal search.

6.2 Multi-Goal iRASST

One final modification to the planning algorithm is selected to promote faster convergence and improve finite-time performance. Initially, the landmark risk was treated independently to the heuristics for reaching the goal node. However, this causes flight patterns that miss landmarks and may run into local optima or risk thresholds. Therefore, the decision is made to modify the landmark visits to be forced through multiple goals. A list of all the landmarks, the original goal defined, and the start node are joined together to define the positions in which viewpoint risk must be minimized, and a minimum clearance must be achieved. The Return-To-Home (RTH) feature is included by the landmark visit queue ending with the start location, as shown in Equation 6.1. This does limit the accuracy of the optimal path by flying to the nearest landmark and following a pattern based on what landmarks still require knowledge acquisition. However, the solution converges much faster than the iRASST algorithm because the initial maneuvers target partial paths that reduce viewpoint risk first and then look to find better paths when it comes to energy and collision risk. The algorithm is referred to as Informed Disaster Risk-Aware SST, iDRASST.

$$\mathcal{L} = [L_1, L_2, \dots, L_{n-1}, G, S] \quad (6.1)$$

During flight, the landmark data is approximated with the knowledge function defined in Chapter 5. In addition, the Pix2Pix urban feature predictions from Chapter 3 are used

to track the flight environment and confirm the landmark is the expected landcover type. After human-robot coordination is integrated, this information can all be relayed to a first responder.

Results for a preliminary demonstration is shown in Figure 6.2. The scenario features a large-scale, mostly empty 3D environment, and therefore the landmarks are found rather quickly at the start of the flight. This leads to the risk factor rapidly decreasing down to about 3. However, the quickest flights to reach the goal and then return home to the start location require close maneuvers with obstacles. This leads to two instances of 9-10 risk values. As a demonstration of all three risks, the final path to return to the start location is initiated from over 1000 meters away, resulting in a large uncertainty of the energy use over the flight. Furthermore, the large uncertainty leads to a large jump in risk that approaches the max value of 10. The total predicted energy remains around 50%, and therefore, the uncertainty of the final flight maneuver would lead to alternative path options if the algorithm continued to search.

The results are stochastic and change based on user inputs for the scenario. These inputs, referred to as *knobs*, allow the user to change system-level assumptions or environmental factors. For example, the viewpoint risk may be inaccurate for a UAS with a smaller camera onboard, and therefore the distance and scaling factors for the viewpoint knowledge function are changed. Alternatively, the scenario of interest could be just after a devastating storm, and the wind and gust parameters need to be scaled to higher and more uncertain wind speeds. These parameters are all available for users or stakeholders to modify before simulation.

6.3 Final Demonstration

The SAFER framework is completed with a set of rapidly generated maps, kinodynamic planning algorithms, and risk-aware optimization. The software environment is prepared with visualization and verification tools that allow users and stakeholders to investigate

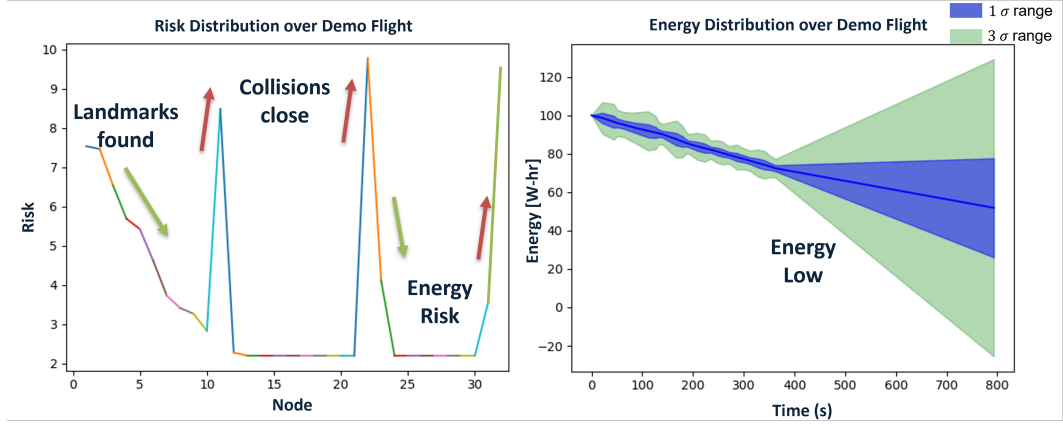


Figure 6.2: Final Demo in Randomly Generated Map

scenarios, tune parameters, and explore *knobs* that can change the results and produce insight. The framework requires a dynamic system model, used as a black-box function, and geospatial datasets, assumed to be open-source. The planning, data, and mapping algorithms process the data and models in a scenario of interest and use the model, algorithm, and environment parameters provided. Single or repeated demonstrations are then possible through the SAFER framework, which outputs visualizations of the map and flight, detailed trajectory information with system and environment states, and risk metrics for energy, collisions, viewpoint knowledge of landmarks, and essentially the risk of mission failure. The framework is represented in Figure 6.5 and is used in this manner for the final demonstration.

The mapping and planning algorithms are demonstrated within the Atlanta flood zone map. The visualization and performance of the iDRASST algorithm are shown in Figure 6.3 and Figure 6.4 respectively. The change of risk over the flight changes based on the three risk factors: energy, collision, and viewpoint. The resulting path is visualized in Google Earth Engine to demonstrate the ability of the framework to model real-world flight paths. The results show that the iDRASST algorithm also performs successfully for faster convergence and safer flight paths. Safer flight paths are defined by lower energy use, being generally far enough away from obstacles, and gaining situational awareness through

viewpoint models of key landmarks. The iDRASST algorithm forces all landmarks to be successfully detected and reduced to zero risk before returning to home. The demonstration in Figure 6.3 is the optimal flight path over a set of simulations run under windy and noisy scenarios. The total energy use for flight is over 50 Watt-hours, the flight time is around 20 minutes, and the mean risk is approximately 4. The max risk of 10 is met at the start, caused by viewpoint risk, and when the flight path goes close to structures at low-altitude. The final risk of 5 is caused by moving towards the goal, but also getting close to structures at a low-altitude. As in previous research, the takeoff and landing maneuvers may need to be added as subroutines within the algorithm through start and goal edge constraints.



Figure 6.3: Visualization of Path for Atlanta, GT Campus Flood Zones

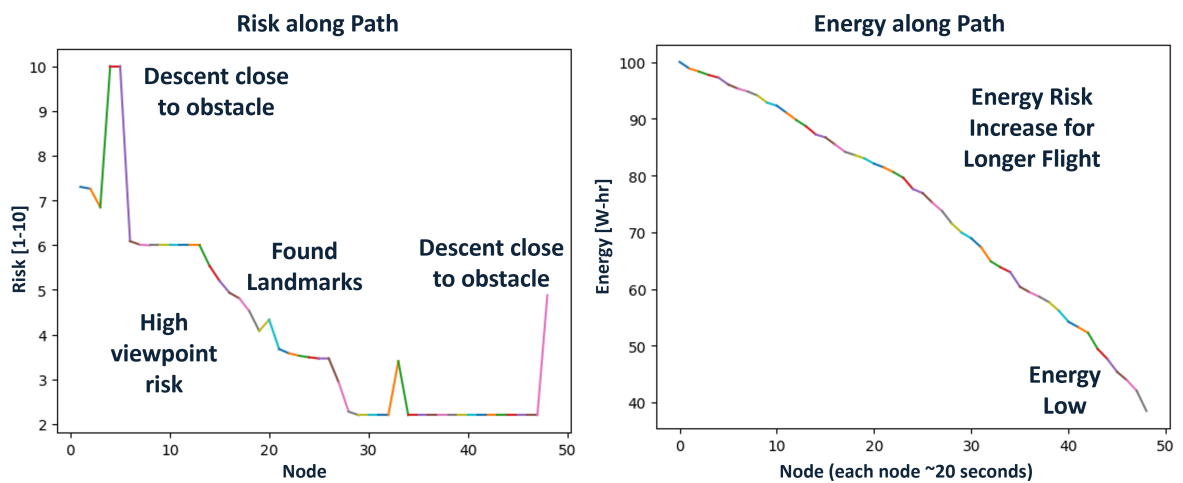


Figure 6.4: Risk and Energy for Path

For disaster response applications such as the example in Atlanta, the system and sensor performance is critical for success, as clearly shown by the risk metrics. This introduces another capability of the SAFER framework, which is to use simulated scenarios to map mission performance to system, or systems of systems, design variables. For example, the energy use required to detect all the landmarks in Figure 6.4 is quickly approaching high-risk levels below 40%, and therefore similar missions may require increased energy availability through a more efficient aerial system or larger battery. Note that the current model is a quadcopter, and therefore a fixed-wing unmanned system may demonstrate improved behavior as long as viewpoint knowledge acquisition can be assumed similarly successful to the quadcopter model. In addition, the takeoff position, or alternative takeoff and landing positions, maybe more useful when multiple operators or multiple aerial systems are in use. Experiments that seek to optimize system performance or concept of operations design are left for future research.

6.4 Summary and Future Work

The iRASST algorithm is demonstrated in the multi-goal setting to examine the effectiveness of performing complete Return-To-Home routes that simultaneously avoid the risk of mission failure and gain viewpoint information about landmarks. The performance shows improvement over traditional methods and is compared directly to the SST algorithms from Chapters 4 and 5. The framework could be great for disaster response teams to deploy autonomous aerial systems for situational awareness. If real-time behaviors are integrated, the aerial systems can be enlisted to collect data while avoiding energy-constraining and obstacle-dense paths. Therefore, guaranteeing a successful data collection mission without needing hardware-in-the-loop simulation, but instead only offline trajectory planning in the SAFER framework.

Faigl in [68] explores many different techniques and algorithms to implement multi-goal planning that improves upon the traditional Traveling Salesman Problem for 3D tra-

jectory planning applications. In addition, the environment could be used to investigate onboard data visualization for safe flight planning in Human-Robot coordination. Future work could benefit from leveraging tools to provide in-flight, real-time feedback, which will require flight testing and sensor tuning.

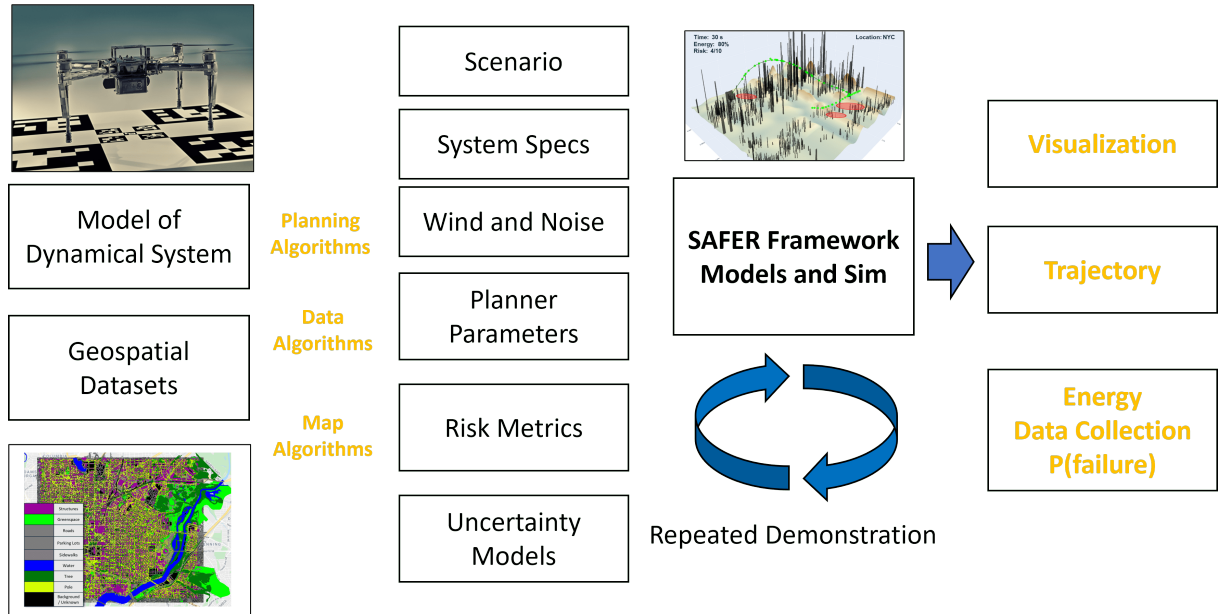


Figure 6.5: Overview of Final Demonstration Experiment

CONCLUSION AND FUTURE WORK

Disaster response missions are dynamic and dangerous events for first responders, and active situational awareness is critical for effective decision-making. In recent years, the technological advancements of unmanned aerial assets have successfully extended the range and output of visual sensors. Aerial assets have demonstrated their capability in disaster response missions via decentralized operations. However, the current market lacks the technology infusion to quickly and effectively integrate UAS tightly into the response team. Moreover, literature lacks a systematic investigation of the algorithms, datasets, and tools for aerial system trajectory planning in urban disasters that optimize mission performance and guarantee success.

This work develops a framework and software environment to investigate the requirements for offline planning algorithms and flight risk models when applied to aerial assets exploring urban disaster zones. The framework is constructed by creating rapid urban maps, demonstrating efficient flight planning algorithms, and integrating formal risk measures that are demonstrated in scenario-driven experiments and Monte Carlo simulations. First, rapid urban mapping strategies are compared for efficient processing and storage through independent obstacle and terrain layers. Open-source data is used when available and is supplemented with an urban feature prediction model trained on satellite imagery using deep learning. Second, sampling-based planners are evaluated for efficient and effective trajectory planning of nonlinear aerial dynamic systems. The algorithms are shown to find collision-free, kinodynamic feasible trajectories using stochastic open-loop control models. Alternative open-loop control commands are formed to improve the planning algorithm's speed and convergence. Third, a risk-aware implementation of the planning algorithm is developed that considers the uncertainty of energy, collisions, and onboard viewpoint data

and then maps them to a single measure of the likelihood of mission failure.

7.1 Summary of Research Questions

Three modules are combined in a framework where the rapid urban maps and risk-aware planner are evaluated against benchmarks for mission success, performance, and speed while creating a unique set of benchmarks from open-source data and software. The three modules address the three core research questions raised during the thesis.

Research Question 1

What data structures and machine learning algorithms successfully generate urban map models with open-source data and tools for disaster response with field robotics?

The hypotheses for research question 1 are summarized as follows.

- Hypothesis 1.1: Independent layers form accurate urban maps and individual layer algorithms require a software framework to evaluate accuracy and speed for unique scenarios
- Hypothesis 1.2: Data-driven models supplement the lack of data in open-source datasets for structures and other urban landcover labels

Two experiments address the question through the Rapid Urban Map, RUM, framework that produces three example maps for Los Angeles, New York City, and Atlanta. The first assumption is that four independent layers define the urban map for urban disaster trajectory planning. The four layers are terrain, structures, weather, and urban landcover features. Terrain and structures define the 3D spatial information, weather defines the flight behavior in different wind and atmospheres, and urban landcover features describe the environment around a flight path. In experiment one, open-source data and Python packages are used in experiments for the terrain approximation and structure discretization. In experiment two, public aerial datasets and TensorFlow tools are used for urban landcover prediction.

The first experiment shows that the rapid urban map module generates a 3D structure and terrain map within 20 meters of truth data and in less than five minutes. The FME Workbench tool performs verification of the accuracy and speed. The Python framework demonstrates that the Gaussian Process, GP, terrain model performs better than B-spline and NURBS models in small-scale, mountainous environments at 10-meter resolution. The GP's error to the 1-meter truth dataset is around 30 meters per sample and is generated in about two minutes. However, environments with relatively flat terrain and wider structures generally perform well using the simplest model, B-spline, and a 20-meter or more discretization of the structures that can be generated in under one minute for small to medium maps. GP is the most complex model and performs the best in complex terrain, but a Bayesian model also predicts the variance of the prediction. The hypothesis is proven correct by the two sub-experiments for structure and terrain modeling and the integration with weather and gust data. An urban map model is accurately and efficiently formed after the use of the RUM framework to evaluate the proper terrain model between B-spline, NURBS, and Gaussian Process, and then the efficient structure distribution parameters of resolution and volume threshold.

The second experiment explores supplementary data for structures and other urban landcover features. The Pix2Pix Generative Adversarial Network, GAN, which uses a 3-channel encoding for nine labels, predicts structures, greenspaces, water, and roads with high accuracy according to the F1, IOU, and pixel accuracy metrics. Structure predictions result in over 80% per-class pixel accuracy and over 0.80 for F1 and IOU scores. In addition, greenspace predictions result in almost 95% per-class pixel accuracy and about 0.90 F1 and IOU scores. The evaluation metrics are generated over a subset of the custom dataset that is not used during training. The example map for Atlanta, Georgia, represents a 10-kilometer squared region with over 300 structures, relatively flat terrain, and the five key Pix2Pix urban landcover classes transformed into vector polygons. The storage size is less than 300 MB and can be generated in less than 5 minutes. The hypothesis is proven correct

that the data-driven method supplements the missing data and provides additional features not available in previous methods. All the experiments prove successful and consistent using exclusively open-source data and a single laptop.

Research Question 2

What trajectory planning algorithm consistently and efficiently finds offline kinodynamic planning algorithms in 3D while operating with unknown dynamics in uncertain environments?

The hypotheses for research question 2 are summarized as follows.

- Hypothesis 2.1: Sampling-based planners that leverage black-box dynamics models have a good finite-time performance by leveraging 'informative sparse likely paths'
- Hypothesis 2.2: Computational efficiency of the algorithm for Monte Carlo simulation relies on efficient queries of the dynamics model that is improved by approximating maneuvers from the black-box model

Two experiments demonstrate the sampling-based planner's effectiveness and efficiency with small-scale and large-scale maps and black-box dynamic models. The cost of trajectories after a finite-time experiment, limited by search iterations, is compared between the RRT and SST algorithms. Monte Carlo simulations demonstrate the SST algorithm's ability to find consistently efficient flight paths even with wind speeds of five meters per second, gusts that double the wind speed, and zero-centered Gaussian noise in the acceleration equations of the dynamics.

The first experiment demonstrates why the sampling-based planning algorithm is selected for forming collision-free, 3D offline flight paths with a black-box dynamics model of a quadcopter. Sampling-based planners prove successful for efficient and optimal flight paths through randomly generated and rapid urban maps, even under wind and noise uncertainty. The Stable-Sparse-RRT, SST, algorithm is shown to improve trajectories for minimum Euclidean distance more consistently and efficiently than the RRT algorithm,

with a 50% improvement in finite-time path convergence for large-scale urban maps. The key parameters for the SST algorithm are the dynamic propagation time bounds, T_{min} and T_{max} , as well as the radius values for finding the best node and nearest witness, Δ_{BN} and Δ_S .

The second experiment explores the forward propagation dynamics and details how the black-box model is replaced with a lattice of predefined maneuvers, or motion primitives, that is 5-15 times more computationally efficient. The motion primitives are generated using an inverse lower-order dynamics model to track motor speeds and the Differential Dynamic Programming, DDP, algorithm to iteratively improve the final position and energy use of the path.

Research Question 3

How can risk measures be incorporated into trajectory planning algorithms to gain confidence for safe and informative flight in uncertain environments?

The hypothesis for research question 3 is summarized as follows.

- Hypothesis 3.1: A formal analysis of the most important risk metrics and the use of a single statistical measure of risk in the planning algorithm measures the safety of flight maneuvers leading to more robust and informative trajectories. Risks include energy use in flight, the distance to collision with an obstacle, and the data collected for first responders at special *landmarks*.

The same maps and dynamic models are demonstrated for experiment 3. However, three trajectory planning algorithms that incorporate risk into the cost function are compared to the original SST algorithm for flight time, energy, and average risk. Energy-aware SST, EASST, incorporates the risk metric as a Gaussian random variable tracked with a Kalman Filter, and the informed Energy-Aware SST, iEASST, adds heuristics to make better decisions for adding edges that lead to faster convergence to near-optimal paths. The informed Risk-Aware SST, iRASST, algorithm leverages all three risk metrics and must

balance energy, collision, and viewpoint risk along the flight. Situational awareness and collision performance under uncertainty is constrained within tight bounds when using iRASST.

The risk-aware planning algorithm generates optimal paths based on three risk metrics of energy, collision, and viewpoint risk and quantifies the likelihood of worst-case events using the Conditional-Value-at-Risk, CVaR, metric. The sampling-based planning algorithm is improved with informative paths, and three versions of the algorithm are compared for the best performance in different scenarios. Energy risk added to the planning algorithm results in 5-35% energy reduction and 20-30% more consistency in finite-time convergence for flight paths in large-scale urban maps. All three risk metrics in the planning algorithm generally result in more energy use than the planner with only energy risk but reduce the mean flight path risk by 10-50% depending on the environment, energy available, and viewpoint landmarks. In summary, flight paths result in less energy when using iEASST, and if collision and viewpoint risk are necessary, iRASST reduces the overall risk of the flight. A playbook outlines the appropriate algorithm in specific scenarios

The three hypotheses and experiments lead to conclusions for the overarching research objective.

Overarching Research Objective

Perform a systematic investigation of the best data and planning algorithms through current technology and tools for unmanned aerial systems to supplement first responders' situational awareness.

A final experiment in an Atlanta flooding scenario demonstrates the framework's full capability with the rapid urban map displaying essential features and the trajectory planner reporting flight time, energy consumption, and total risk. Furthermore, the simulation environment provides insight into offline planning limitations through Monte Carlo simulations with environment wind and system dynamics noise. The iRASST algorithm is modified to the Informed Disaster Risk-Aware SST, iDRASST, algorithm by replacing the single land-

mark search list with a queue of landmark goal points. The viewpoint risk is updated the same as before. However, the updated algorithm requires all landmark targets to be found before heading towards the final goal, which is the original start location. Therefore, the algorithm incorporates the necessary Return-To-Home functionality. The algorithm improves the consistency of iRASST by 50% when landmarks are not along the path or are challenging to detect.

7.2 Summary of Contributions

This dissertation outlines a methodology and provides a framework for experimenting with aerial systems for situational awareness during disaster response missions. The complete list of contributions is summarized here.

7.2.1 Framework and Tools

A complete framework and software environment is developed in python, with modules that each operate as an independent framework for mapping, planning, and risk-aware planning, respectively. The Rapid Urban Map, RUM, framework allows for algorithm comparisons for terrain, structures, and wind models. Unique class structures that can be used by internal tools and the trajectory planning class are available to use or expand. The sampling-based planner framework starts with a first principles tree-based planner and expands from RRT and SST to the risk-aware planning algorithms introduced. The risk metrics and the CVaR measure are defined to calculate risk values based on the different planner and dynamic model inputs, depending on which risk metric is used. All flight performance and risk metrics are stored along the flight and can be visualized in one of two visualization platforms, Matplotlib and Plotly. The visualization tools allow for key metrics to be visualized while verifying constraints for terrain and structures. An overview of the software environment is shown in Figure 7.1.

All framework modules integrate with open-source data sets and public software pack-

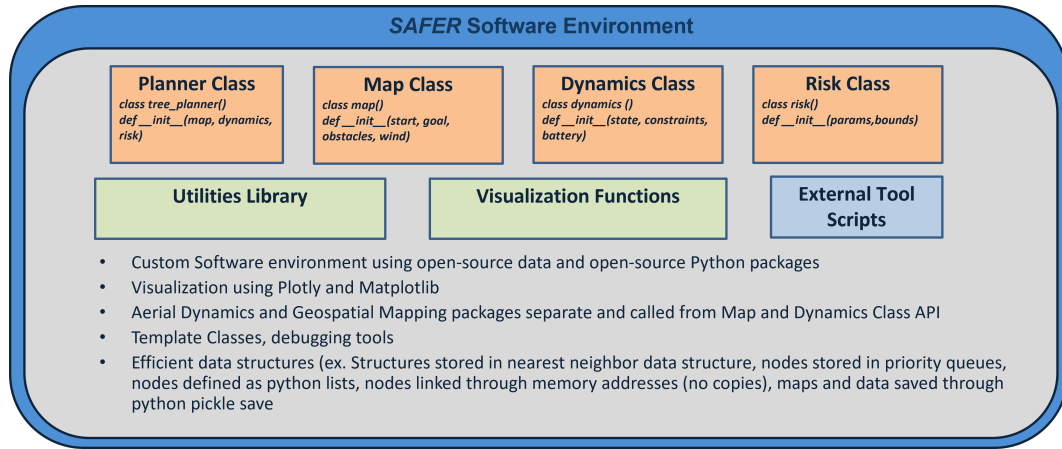


Figure 7.1: The SAFER Software Environment

ages. External tools like FME Workbench and Google Earth are used for statistical and visual verification and validation. Google packages and tools like TensorFlow, Colab, and Earth Engine are used for the RUM's Pix2Pix data creation, training, visualization, and transformation.

7.2.2 Benchmark and Demonstrations

A series of experiments are outlined through the chapters that lead to new or alternative benchmarks for the three modules and research questions. The mapping and planning tasks each have unique *playbooks* to use for deciding the best algorithm to apply for a scenario. For instance, the B-spline algorithm is suggested for regions that are not mountainous and are large in size. Another example is that the iEASST algorithm is suggested for scenarios where energy is the most critical metric and fast convergence is important. Overall, four unique planning algorithms are generated during the planning chapters. The Stable-Sparse RRT, SST, algorithm is upgraded to the EASST, iEASST, iRASST, and iDRASST algorithms. Each algorithm is evaluated for its pros and cons of use.

The complete framework operates as a preflight planning and simulation tool that could function for active response planning or for investigating system and system of systems capabilities during simulated events. This is demonstrated in a flooding scenario in Atlanta,

and Figure 7.2 shows the final demo flight path in Google Earth for verification. The framework and software environment are made available to use as benchmarks in the field to serve as a foundation for increasing first responders' safety in the challenging task of urban disaster response.



Figure 7.2: Final Demonstration Flight Waypoints in Atlanta, Visualized in Google Earth

7.3 Next Steps and Future Work

The experiments demonstrate a methodology for aerial disaster response and situational awareness that is capable with current technology and a single laptop in the field. Therefore, flight experiments could be implemented to continue the validation of trajectory performance and risk measures. However, real-world use in disaster scenarios, rather than academic settings, does require more advanced communication and a mobile operating base that needs to be investigated. Specific items for improvements to the modules' algorithms or software framework are detailed at the end of the previous chapters. However, a few of the most important future work topics are introduced and explained.

Additional risk models have been leveraged in the past, such as in [49], that include

localization error models using GPS satellites and Geometric Dilution of Precision, GDOP. In addition, communication issues could be modeled to account for communication back with first responders. For example, work in [181] investigated UAV positioning in an obstructive environment, where buildings and trees can disrupt communication signals. Communication maps were created for modeling communication risk in [182]. In addition, FAA regulations that were not considered in this work could be integrated into the collision or viewpoint risk metrics depending on the limitations, or they could be added as hard constraints in the flight zones that indicate feasible flight states. Chapter 1 claimed that more autonomous or semi-autonomous behavior is needed for aerial system integration with first responders. For the SAFER framework planning algorithm to be effective, the aerial systems must be capable of autonomous and Beyond Visual Line of Sight, BVLOS, flight. However, the level of autonomy, or semi-autonomy, still needs to be determined for optimal human-robot interaction with first responders.

For trajectory planning, alternative methods to sampling-based could be explicitly compared in the new benchmarking environment with rapid urban maps to determine the capabilities and limitations as compared to alternative methods, such as Mixed Integer Linear Programming, MILP. Multi-agent planning should be considered as the organization and coordination of the multiple aerial systems in use can provide additional insight into the optimal navigation or risk in the area, therefore requiring a plan of operations and coordination of the multiple systems as in [183] and [184]. Lastly, new simulation environments that improve the dynamics, collision, and sensor models could be used for simulations with high-fidelity dynamics and graphic rendering engines. This step can lead to physical experiments with real aircraft to gain insightful knowledge on real-time planning constraints.

Appendices

APPENDIX A

ADDITIONAL THEORY

A.1 Probability and Statistics

The fundamentals of probability theory are assumed throughout the dissertation. In particular, the assumption of Gaussian random variables is used in every chapter whether for machine learning, uncertainty modeling, or energy estimation. A Gaussian, scalar random variable, X , is defined as having a probability density function $p(x)$ as in Equation A.1. Gaussian random variables are commonly deployed for two reasons. One, the central limit theorem established that when independent random variables are combined the normalized sum is approximately normally distribution. More specifically, the sample means approximate a Gaussian distribution more and more as the sample size of random variables gets larger. Two, the Gaussian random variable has many properties that make mathematical formulas simple and closed-form. The Gaussian random variable can be defined by the two parameters shown in Equation A.1 and the expectation is equal to the sample mean. Let X be a scalar random variable with a probability density function $p(x)$ and the function of interest be $f(x)$. The general formula for the expectation of a random variable is in Equation A.2.

$$p(x) = \frac{1}{\sqrt{2 * \pi} \sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \quad (\text{A.1})$$

$$\mathbb{E}[f(X)] = \int_{-\infty}^{\infty} f(x)p(x)dx \quad (\text{A.2})$$

In Chapter 5, the risk distributions are investigated as to whether or not they can be approximated as Gaussian random variables. It is discovered that when high risk values

are encountered the Gaussian or Weibull distributions fail to catch the tail-end of the distribution. Addition details are shown here by examining the distributions using the python package Fitter. The package is used to demonstrate how different distributions fit to the sampled Gaussian distribution and the resulting risk distribution. Figure A.1 visualizes the fits for the following distributions: Gaussian, gamma, Log-normal, Weibull, Rayleigh, uniform, exponential. Weibull performs better than Gaussian in catching tail-end risks as shown in Figure A.1.

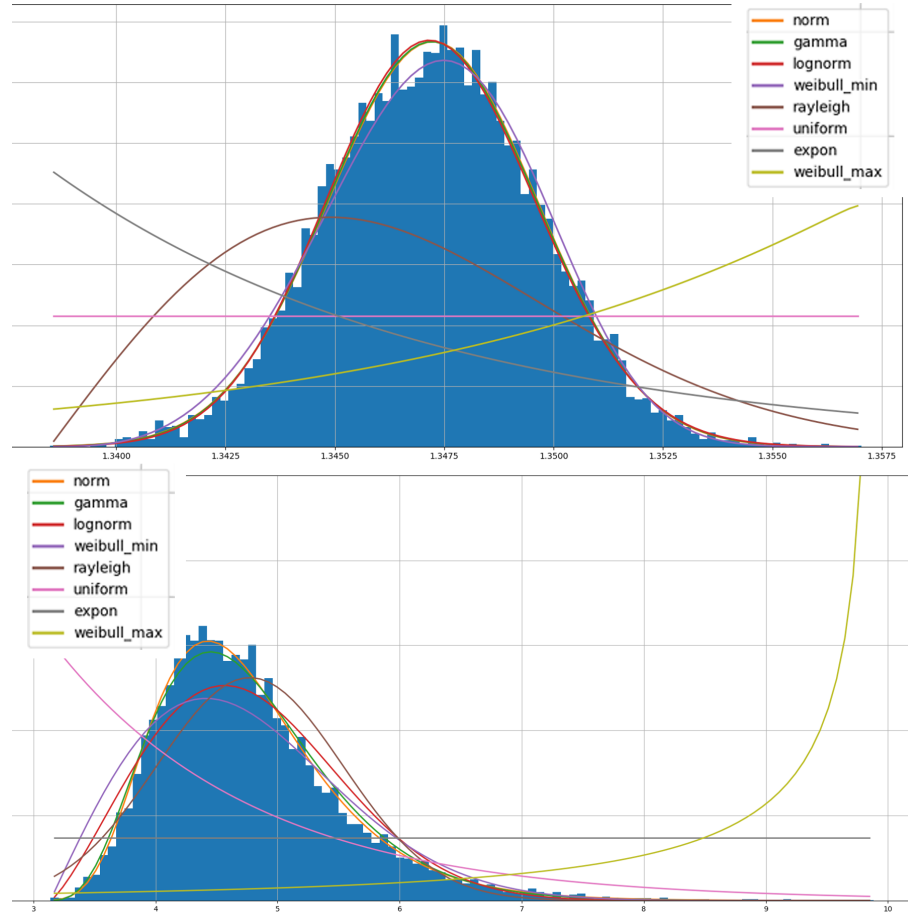


Figure A.1: Fitting Data to Distribution, Visualization with the Python Package Fitter

A.2 Weibull Distribution and CVaR Formula

The Weibull distribution has parameters, shape, scale, and location. The location parameter is a fallout from the other variables, therefore the shape, k , and the scale, λ , are only needed.

The random variable is $X \sim Weibull(\lambda, k)$, where the parameters are both real numbers greater than zero. From probability theory the first and second moments, or expectation and variance respectively, are as $E[X] = \lambda\Gamma(1 + \frac{1}{k})$ and $\sigma^2(X) = \lambda^2[\Gamma(1 + \frac{2}{k}) - \Gamma(1 + \frac{1}{k})^2]$ where the gamma function is $\Gamma(a) = \int_0^\infty p^{a-1}e^{-p}dp$. The CDF and PDF are then

$$F(x) = 1 - e^{-(x/\lambda)^k}$$

$$f(x) = \begin{cases} \frac{k}{\lambda} (\frac{x}{\lambda})^{k-1} e^{-(x/\lambda)^k} & x > 0 \\ 0 & x < 0 \end{cases}$$

The VaR and CVaR formulas are used for risk mapping of Weibull and Gaussian distributions. The formulas for the Weibull distribution is shown in the following equations.

$\Gamma_U(a, b) = \int_b^\infty p^{a-1}e^{-p}dp$ is the upper incomplete gamma function.

$$VaR(X) = q_\alpha = \lambda(-\ln(1 - \alpha))^{1/k}$$

$$CVaR(X) = \bar{q}_\alpha = \frac{\lambda}{1 - \alpha} \Gamma_U(1 + \frac{1}{k}, -\ln(1 - \alpha))$$

An alternative to using the Gaussian or Weibull CVaR formulas is to derive the CVaR formula directly for the risk function. A partial solution to the derived expectation formula for the energy risk is shown here, as a start to the full CVaR derivation. The work can be expanded to find the closed-form solution to the CVaR in future work (hint: integration by parts). Take the fact that the random variable X is Gaussian, $X \sim \mathcal{N}(\mu, \sigma^2)$. For the risk function defined for energy risk, $G(X)$, and the random Gaussian variable with normal distribution, $p(x)$, the expectation is solved in Equation A.3.

$$\mathbb{E}[f(X)] = \int_{-\infty}^{\infty} \exp\left(\frac{\gamma}{\max\{x - b, \lambda\}}\right) \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(x - \mu)^2}{2\sigma^2}\right) dx \quad (\text{A.3})$$

Using the exponential product rule, the property that constants can move out of the integral and the domain of x through the $\max()$ function, results in the following formula.

$$\mathbb{E}[f(X)] = \frac{1}{\sqrt{2\pi}\sigma} \left(\int_{x_L}^{\lambda} \exp\left(\frac{\gamma}{\lambda} - \frac{(x - \mu)^2}{2\sigma^2}\right) dx + \int_{\lambda}^{x_u} \exp\left(\frac{\gamma}{x - b} - \frac{(x - \mu)^2}{2\sigma^2}\right) dx \right)$$

A.3 Aerial Planning and Control

Background information that is used when experimenting with the aerial dynamics model and comparing trajectory planning algorithms is detailed in this section. For one, background into optimal control techniques is important when deciding to approach the problem alternatively with sampling-based planning. In addition, linear control theory is used when evaluating motion primitive stability.

Optimal control problems for continuous systems and constraints require a formal set up to prepare to solve. The dynamic system is assumed a set of differential equations such that $\dot{x} = f(x, u, t)$ and with boundary conditions for the state, x , and time, t . the objective is to minimize the performance index, or objective function, which is a combination of a running and final cost.

$$J = \Phi(x(t_f), t_f) + \int_{t_0}^{t_f} L(x(t), u(t), t)$$

Preparing to solve this constrained optimization problem leads to the formation of the Hamiltonian, with two paths forward. Sticking with the continuous realm, an open-loop or closed-loop assumption can be made resulting in either the Pontryagin's Minimum Principle or the Hamilton-Jacobi Bellman Equations. The HJB partial differential equations are difficult to solve because of the high non-linearity and strict constraints. A discrete approach can be taken, leveraging dynamic programming methods to form the Bellman equations. However, these suffer from the curse of dimensionality of high-dimensional

state spaces. Advanced methods improve upon this through shooting or direct collocation methods or by framing the problem to approximate the optimal solution.

One solution to this is to assume the dynamics to be linear and formulate the Linear Quadratic Regulator (LQR) problem. The LQR problem requires solving the Algebraic Ricatti Equation, which has a wide range of accurate, computationally efficient methods in literature.

$$\begin{aligned} \text{Continuous : } & A^T P + P A - P B R^{-1} B^T P + Q = 0 \\ \text{Discrete : } & P = A^T P A - (A^T P B)(R + B^T P B)^{-1}(B^T P A) + Q \end{aligned} \tag{A.4}$$

A.4 Linear System Stability

First, a linear-time-invariant (LTI) system with state feedback control as in Equation A.5 is assumed.

$$\dot{x} = Ax + Bu \quad u = Kx \tag{A.5}$$

From linear system theory [185] it is known that a system can be proven to be stable by defining a Lyapunov function, $V(x)$, that holds from three facts.

- $V(0) = 0$
- $V(x) \geq 0 \quad \forall x \neq 0$
- $\dot{V}(x) \leq 0 \quad \forall x \neq 0$

Let the Lyapunov function be defined as $V = x^T P x$ where P is a symmetric matrix parametrized by time, t , so that $P(t) = P^T(t)$. The derivation of the optimal closed loop feedback control is as follows.

$$\dot{V}(x) = \frac{\delta}{\delta t} V(x) = \frac{\delta}{\delta t} [x^T(t)P(t)x(t)] = \dot{x}^T(t)P(t)x(t) + x^T(t)P(t)\dot{x}(t) + x^T(t)\dot{P}(t)x(t)$$

$$\text{Take } \dot{x}(t) = Ax(t) + Bu(t) = (A + BK(t))x(t)$$

$$\dot{V}(x) = \{(A + BK)x\}^T Px + x^T P\{(A + BK)x\} + x^T \dot{P}x$$

$$\dot{V}(x) = x^T \{A^T P + PA + K^T B^T P + PBK + \dot{P}\}x$$

For the Lyapunov equation to hold, the equation must be equal or less to zero. Therefore, a matrix Q that is positive-semi-definite is assumed to exist.

$$\dot{P} = A^T P + PA + K^T B^T P + PBK + Q$$

Assuming the solution to the optimal closed loop control input, the continuous-time differential ricatti equation is the following.

$$\dot{P}(t) = A^T P(t) + P(t)A - P(t)BR^{-1}B^T P(t) + Q$$

As proven in previous works, if the problem is assumed to be an infinite horizon problem, the limit can be taken as $t \rightarrow \infty$, meaning $\dot{P}(t) \rightarrow 0$ and $P(t) \rightarrow P_\infty$. This results in the continuous-time algebraic ricatti equation, CARE, in Equation A.6. Alternatively, the CARE equation can be formed from setting up the same problem as a constrained optimization problem through the Hamiltonian, detailed by Bertsekas in [186].

$$A^T P + PA - PBR^{-1}B^T P + Q = 0 \tag{A.6}$$

If linearized dynamics are considered, $\dot{x} = Ax(t) + Bu(t)$, then the forward reachable

sets can be defined as quick and efficient calculations shown in Equation A.7 [116], or from hyper-sphere approximations shown in Equation A.8 [187]. This integration is left for future work.

$$\mathcal{R}_f[t_0, t] = \{\mathbf{x} \in \mathcal{X} \mid \forall \mathbf{u} \text{ s.t. } \mathbf{x} = e^{-A(t-t_0)}x_s - \int_{t_0}^t e^{-A(t-\tau)}Bu(\tau)d\tau\} \quad (\text{A.7})$$

$$\mathcal{F}[t_0, t] = \{x \in \mathcal{X} \mid \|x - e^{A(t-t_0)}x_s\|_2 \leq r(t_0, t, u_{max})\} \quad (\text{A.8})$$

APPENDIX B

SUPPLEMENTARY RESULTS

B.1 Stable Sparse RRT Detailed Results

Tables documenting the results from comparing RRT and SST, then comparing iEASST and SST, are included in the following section. Table B.1 shows the detailed results from Chapter 4’s comparison of the RRT and SST algorithms. The performance of the algorithm is compared for changing parameters of sampling radius, dynamic propagation time, wind speed, noise variance, number of obstacles in map, scale of the map, and the number of iterations to run the algorithms. The table shows how the selection of the SST time parameters is important to improving the cost as compared to RRT. SST consistently uses less nodes across all map sizes, however.

Table B.2 compares three of the algorithms from Chapter 5 in four generated 3D maps, with boxplots of the results found in Figure 5.17. The tables provide detailed insight of the key outcomes for adding the energy risk metric to form EASST and in adding informed heuristics to form iEASST. T_{min} and T_{max} are the minimum and maximum times for SST’s uniform distribution of dynamic propagation time. DW is the witness radius, Δ_W , and DS is the sampling radius, Δ_S . A is the acceleration noise standard deviation and W is the average wind speed magnitude before adding the Dryden gust model. The algorithms’ performances are measured by the total number of nodes formed, the final node’s risk value (euclidean cost for SST), the total time to compute, the flight energy, and the flight time. The clearest indication of the iEASST algorithm’s success is the flight energy and time values compared to SST on map 14. In addition, the flight performance of iEASST does not seem directly affected by wind and noise, as the algorithm finds alternative flight maneuvers that result in efficient paths.

Table B.1: RRT vs. SST for Random 3D Map Trajectory Search, Empty cells indicate searches that failed to find a path

Iterations	num_obstacles	map_scale	sst_time_low	sst_time_high	rtr_time	delta_witness	delta_sample	wind_speed (m/s)	noise_sigma (m/s ²)	rtr_nodes	rtr_cost (m)	rtr_runtime (s)	sst_nodes	sst_cost (m)	sst_runtime (s)
557	10	1	2.85	9.45	9.45	9.02	2.66	4.42	0.21	294	90.43	25.33	129	153.59	27.79
762	10	1	3.33	9.28	9.28	13.15	3.47	1.07	0.10	292	113.64	27.62	118	116.17	29.50
3843	10	1	3.58	10.22	10.22	10.73	3.72	5.00	0.11	1363	109.85	163.82	450	114.32	253.94
2077	10	1	2.24	3.67	3.67	14.51	3.91	3.56	0.36	625	125.59	32.97	29	122.27	73.02
4692	10	2	4.34	5.86	5.86	11.85	3.48	3.46	0.14	3503	303.01	111.84	1053	294.40	214.63
2440	20	1	2.87	13.49	13.49	13.98	4.41	4.81	0.46	484	101.44	169.86	183	98.67	173.67
3971	20	1	4.76	12.72	12.72	11.16	3.80	1.30	0.30	962	107.15	210.30	354	100.90	239.82
655	20	1	4.37	6.68	6.68	13.31	4.90	1.35	0.58	411	108.82	18.47	83	104.99	36.68
4665	20	1	3.61	5.82	5.82	11.47	4.09	1.40	0.71	2373	101.74	121.79	284	104.99	196.87
1450	20	2	3.78	7.44	7.44	7.01	2.95	4.11	0.30	1079	253.06	42.91	732	244.75	72.80
4796	20	2	4.21	10.87	10.87	13.38	5.19	1.73	0.08	2883	251.65	288.47	1016	225.02	385.68
4397	50	1	4.83	15.00	15.00	12.51	3.87	4.61	0.58	751	96.37	247.30	265	106.65	301.14
1078	50	1	2.07	5.72	5.72	14.23	2.68	4.18	0.31	606	116.83	24.52	72	136.86	68.74
2008	50	1	3.25	5.24	5.24	6.23	3.60	1.71	0.36	1070	110.12	46.11	599	126.80	82.39
3009	50	2	3.84	5.68	5.68	9.52	4.86	3.75	0.29	2525	272.76	74.76	989	259.47	142.62
3837	50	3	4.50	5.38	5.38	13.15	3.90	4.09	0.49	3341	394.77	121.67	581	471.21	151.99
681	50	3	4.94	16.18	16.18	11.16	3.96	3.80	0.19	436	389.32	53.05	386	521.29	58.34
618	50	3	3.76	10.85	10.85	5.28	2.60	3.90	0.32	457	337.65	27.95	464	450.61	33.96
3309	100	1	3.46	4.01	4.01	9.76	4.61	3.44	0.55	1808	104.04	61.82	318	111.06	137.25
3860	100	1	4.84	6.10	6.10	9.94	3.86	0.20	0.14	1392	106.65	106.05	512	115.15	263.03
2460	100	1	2.26	12.48	12.48	7.06	3.49	4.41	0.17	279	106.65	124.68	384	104.98	106.96
1536	100	1	2.86	12.42	12.42	6.77	3.00	0.99	0.11	294	51.51	76.94	288	48.46	80.29
4203	100	1	3.57	8.59	8.59	9.13	5.25	1.17	0.29	1248	122.57	152.62	348	106.89	269.08
3137	100	2	2.08	5.07	5.07	6.88	2.45	1.95	0.29	2540	279.56	66.31	1415	279.66	140.98
1544	100	2	3.79	14.16	14.16	8.72	2.53	1.98	0.14	498	228.63	91.68	650	253.60	96.44
2191	100	3	4.76	7.34	7.34	9.30	5.15	0.20	0.68	1739	556.27	69.78	1172	465.19	105.73

Table B.2: SST, EASST, iEASST Performance Comparison

Planner Name	Its	Map	Tmin	Tmax	DW	DS	W	A	Nodes	Risk	Time (s)	Energy [W-hr]	Flight T(s)
EASST	500	14	5	20	5	3	0.97	0.25	117	1.41	20.88	8.23	36.20
EASST	500	14	5	20	5	3	3.70	0.23	116	1.51	21.22	11.56	61.27
EASST	500	14	5	20	5	3	1.10	0.28	149	1.69	21.46	16.30	74.28
EASST	500	14	5	20	5	3	3.35	0.35	147	1.41	21.19	7.41	59.92
EASST	500	14	5	20	5	3	2.75	0.10	143	1.40	20.79	8.56	20.85
EASST	500	14	5	20	5	3	0.79	0.10	156	1.41	22.05	7.92	42.45
EASST	500	14	5	20	5	3	2.12	0.46	117	1.40	22.73	9.19	22.41
EASST	500	14	5	20	5	3	1.26	0.22	160	1.43	22.41	8.54	56.99
EASST	500	14	5	20	5	3	2.22	0.11	150	1.40	22.64	8.08	39.85
EASST	500	14	5	20	5	3	2.06	0.29	90	1.43	22.05	9.76	43.48
EASST	500	14	5	20	5	3	4.14	0.18	122	1.42	21.22	9.29	29.20
EASST	500	14	5	20	5	3	3.98	0.19	79	inf	21.28	0.00	0.00
EASST	500	14	5	20	5	3	2.85	0.08	145	1.41	21.09	9.03	25.58
EASST	500	14	5	20	5	3	0.29	0.12	129	1.38	21.89	6.33	38.48

Table B.2 Continued from previous page

Planner Name	Its	Map	Tmin	Tmax	DW	DS	W	A	Nodes	Risk	Time (s)	Energy [W-hr]	Flight T(s)
EASST	500	14	5	20	5	3	3.56	0.03	124	1.50	22.49	9.81	79.37
EASST	500	14	5	20	5	3	4.00	0.44	87	1.43	21.74	8.33	54.54
EASST	500	14	5	20	5	3	4.26	0.39	126	1.42	21.29	8.79	35.68
EASST	500	14	5	20	5	3	0.37	0.24	143	1.46	20.92	9.87	53.25
EASST	500	14	5	20	5	3	2.30	0.19	148	1.44	21.79	10.34	24.35
EASST	500	14	5	20	5	3	2.11	0.38	157	1.40	22.16	8.21	37.40
EASST	500	14	5	20	5	3	1.57	0.46	113	1.47	21.00	8.56	46.69
EASST	500	14	5	20	5	3	3.33	0.10	114	1.58	20.55	12.78	66.53
EASST	500	14	5	20	5	3	0.89	0.48	147	1.43	20.84	10.81	40.90
EASST	500	14	5	20	5	3	2.92	0.47	133	1.38	21.40	6.73	43.91
EASST	500	14	5	20	5	3	1.75	0.36	139	1.51	21.80	12.82	73.96
EASST	500	14	5	20	5	3	4.40	0.03	111	1.41	21.03	8.67	32.09
EASST	500	14	5	20	5	3	2.55	0.03	142	1.43	21.37	8.35	47.70
EASST	500	14	5	20	5	3	0.96	0.28	146	1.40	21.60	7.76	32.39

Table B.2 Continued from previous page

Planner Name	Its	Map	Tmin	Tmax	DW	DS	W	A	Nodes	Risk	Time (s)	Energy [W-hr]	Flight T(s)
EASST	500	14	5	20	5	3	1.10	0.19	123	1.49	20.64	11.00	70.68
EASST	500	14	5	20	5	3	0.67	0.46	138	1.68	21.57	14.63	88.99
EASST	500	14	5	20	5	3	1.34	0.43	140	1.43	21.33	8.87	36.84
EASST	500	14	5	20	5	3	1.76	0.14	143	1.42	21.03	8.26	43.74
EASST	500	r1-50	5	20	5	3	3.81	0.31	98	1.33	31.97	4.27	11.57
EASST	500	r1-50	5	20	5	3	3.22	0.04	89	1.34	33.25	4.66	15.14
EASST	500	r1-50	5	20	5	3	0.70	0.06	82	1.34	32.10	4.00	18.47
EASST	500	r1-50	5	20	5	3	3.31	0.06	73	1.34	31.61	4.25	15.27
EASST	500	r1-50	5	20	5	3	1.14	0.32	66	1.34	27.18	4.74	24.82
EASST	500	r1-50	5	20	5	3	3.77	0.27	100	1.34	28.38	4.74	19.07
EASST	500	r1-50	5	20	5	3	4.35	0.23	72	1.34	27.23	4.62	10.21
EASST	500	r1-50	5	20	5	3	3.86	0.49	98	1.33	32.12	4.33	20.91
EASST	500	r1-50	5	20	5	3	3.61	0.27	79	1.34	33.12	4.83	5.10
EASST	500	r1-50	5	20	5	3	0.49	0.39	65	1.35	30.84	5.13	6.61

Table B.2 Continued from previous page

Planner Name	Its	Map	Tmin	Tmax	DW	DS	W	A	Nodes	Risk	Time (s)	Energy [W-hr]	Flight T(s)
EASST	500	r1-50	5	20	5	3	2.75	0.27	92	1.34	29.44	4.65	15.53
EASST	500	r1-50	5	20	5	3	0.91	0.48	84	1.34	32.82	4.22	13.46
EASST	500	r1-50	5	20	5	3	3.54	0.03	93	1.34	30.24	4.01	10.70
EASST	500	r1-50	5	20	5	3	2.00	0.44	71	1.34	27.56	5.24	5.22
EASST	500	r1-50	5	20	5	3	0.14	0.28	86	1.33	32.94	4.17	25.46
EASST	500	r1-50	5	20	5	3	0.26	0.11	85	1.34	32.67	3.79	17.41
EASST	500	r1-50	5	20	5	3	3.96	0.37	79	1.34	32.38	4.63	13.11
EASST	500	r1-50	5	20	5	3	0.46	0.32	92	1.34	33.32	4.54	7.97
EASST	500	r1-50	5	20	5	3	1.48	0.08	101	1.35	33.75	5.26	15.61
EASST	500	r1-50	5	20	5	3	4.34	0.24	69	1.34	32.35	4.80	13.98
EASST	500	r1-50	5	20	5	3	1.54	0.38	83	1.33	23.85	3.47	16.62
EASST	500	r1-50	5	20	5	3	2.39	0.06	83	1.33	22.95	3.83	13.93
EASST	500	r1-50	5	20	5	3	4.55	0.42	75	1.34	23.52	3.91	14.01
EASST	500	r1-50	5	20	5	3	2.18	0.29	59	1.33	22.19	4.20	22.27

Table B.2 Continued from previous page

Planner Name	Its	Map	Tmin	Tmax	DW	DS	W	A	Nodes	Risk	Time (s)	Energy [W-hr]	Flight T(s)
EASST	500	r1-50	5	20	5	3	4.70	0.04	73	1.33	22.50	3.80	13.71
EASST	500	r1-50	5	20	5	3	2.68	0.29	77	1.34	22.37	5.14	26.50
EASST	500	r1-50	5	20	5	3	0.86	0.01	84	1.34	22.51	4.32	14.18
EASST	500	r1-50	5	20	5	3	2.09	0.19	77	1.35	22.43	5.35	23.36
EASST	500	r1-50	5	20	5	3	0.67	0.27	77	1.34	22.58	5.13	6.26
EASST	500	r1-50	5	20	5	3	3.76	0.25	105	1.36	26.93	6.04	29.43
EASST	500	r1-50	5	20	5	3	1.67	0.09	63	1.34	21.57	3.80	16.69
EASST	500	r1-50	5	20	5	3	4.29	0.09	92	1.34	21.05	4.56	19.11
EASST	500	r2-100	5	20	5	3	3.79	0.25	163	1.37	40.23	6.57	24.51
EASST	500	r2-100	5	20	5	3	0.96	0.43	198	1.39	32.61	8.22	23.41
EASST	500	r2-100	5	20	5	3	1.75	0.15	191	1.39	29.75	7.54	29.23
EASST	500	r2-100	5	20	5	3	1.97	0.21	187	1.38	29.81	7.45	28.56
EASST	500	r2-100	5	20	5	3	0.81	0.17	195	1.38	34.96	7.83	27.80
EASST	500	r2-100	5	20	5	3	4.45	0.27	194	1.37	40.18	5.52	29.06

Table B.2 Continued from previous page

Planner Name	Its	Map	Tmin	Tmax	DW	DS	W	A	Nodes	Risk	Time (s)	Energy [W-hr]	Flight T(s)
EASST	500	r2-100	5	20	5	3	3.16	0.18	173	1.37	32.53	7.17	28.73
EASST	500	r2-100	5	20	5	3	0.47	0.29	182	1.37	29.08	6.10	26.93
EASST	500	r2-100	5	20	5	3	2.56	0.37	179	1.39	29.74	8.82	44.70
EASST	500	r2-100	5	20	5	3	4.57	0.20	192	1.38	29.66	7.54	30.62
EASST	500	r2-100	5	20	5	3	0.81	0.18	210	1.38	44.52	6.47	32.76
EASST	500	r2-100	5	20	5	3	3.62	0.41	197	1.41	29.06	9.01	39.61
EASST	500	r2-100	5	20	5	3	3.53	0.02	187	1.39	26.18	7.14	33.24
EASST	500	r2-100	5	20	5	3	4.09	0.21	188	1.37	25.68	6.01	30.04
EASST	500	r2-100	5	20	5	3	4.97	0.47	168	1.40	25.38	7.61	51.53
EASST	500	r2-100	5	20	5	3	1.35	0.21	173	1.39	25.79	7.27	40.76
EASST	500	r2-100	5	20	5	3	2.58	0.40	210	1.39	27.44	7.50	30.71
EASST	500	r2-100	5	20	5	3	0.60	0.41	178	1.38	34.66	7.26	39.28
EASST	500	r2-100	5	20	5	3	2.12	0.22	233	1.42	33.77	10.08	25.33
EASST	500	r2-100	5	20	5	3	4.16	0.22	161	1.36	25.33	5.86	33.23

Table B.2 Continued from previous page

Planner Name	Its	Map	Tmin	Tmax	DW	DS	W	A	Nodes	Risk	Time (s)	Energy [W-hr]	Flight T(s)
EASST	500	r2-100	5	20	5	3	3.00	0.28	214	1.37	25.66	6.89	30.79
EASST	500	r2-100	5	20	5	3	2.62	0.19	161	1.39	24.34	7.35	31.98
EASST	500	r2-100	5	20	5	3	2.49	0.07	183	1.39	24.27	7.52	28.60
EASST	500	r2-100	5	20	5	3	0.03	0.19	180	1.35	25.52	3.93	29.97
EASST	500	r2-100	5	20	5	3	1.23	0.25	193	1.38	25.32	7.26	34.76
EASST	500	r2-100	5	20	5	3	4.20	0.05	189	1.38	25.05	7.04	35.22
EASST	500	r2-100	5	20	5	3	0.21	0.24	203	1.37	25.19	5.28	33.03
EASST	500	r2-100	5	20	5	3	0.81	0.46	187	1.42	24.84	10.76	6.71
EASST	500	r2-100	5	20	5	3	2.30	0.18	197	1.40	24.78	9.38	15.34
EASST	500	r2-100	5	20	5	3	3.11	0.38	193	1.38	24.78	8.98	43.30
EASST	500	r2-100	5	20	5	3	3.83	0.02	193	1.39	24.93	7.87	37.39
EASST	500	r2-100	5	20	5	3	3.87	0.39	170	1.40	24.28	9.69	12.30
EASST	500	r2-100	5	20	5	3	3.64	0.45	154	1.37	24.27	7.80	14.87
EASST	500	r2-100	5	20	5	3	4.07	0.38	189	1.39	24.47	8.03	41.95

Table B.2 Continued from previous page

Planner Name	Its	Map	Tmin	Tmax	DW	DS	W	A	Nodes	Risk	Time (s)	Energy [W-hr]	Flight T(s)
EASST	500	r3-200	5	20	5	3	3.36	0.41	268	1.45	31.27	9.56	50.64
EASST	500	r3-200	5	20	5	3	2.35	0.12	242	1.48	32.62	12.09	48.02
EASST	500	r3-200	5	20	5	3	4.63	0.49	233	1.44	32.97	10.63	36.07
EASST	500	r3-200	5	20	5	3	2.66	0.17	244	1.45	29.54	12.25	27.63
EASST	500	r3-200	5	20	5	3	1.73	0.48	254	1.50	31.48	14.02	22.58
EASST	500	r3-200	5	20	5	3	4.94	0.04	227	1.46	33.69	10.30	57.91
EASST	500	r3-200	5	20	5	3	4.34	0.40	217	1.46	31.35	10.37	49.70
EASST	500	r3-200	5	20	5	3	0.81	0.24	282	1.45	32.69	10.14	48.81
EASST	500	r3-200	5	20	5	3	3.15	0.04	250	1.43	30.37	11.05	29.71
EASST	500	r3-200	5	20	5	3	1.71	0.15	236	1.49	30.45	14.34	21.68
EASST	500	r3-200	5	20	5	3	4.23	0.06	236	1.44	30.26	9.56	61.67
EASST	500	r3-200	5	20	5	3	3.09	0.13	245	1.45	30.87	9.68	44.28
EASST	500	r3-200	5	20	5	3	3.89	0.34	256	1.42	32.34	9.39	43.95
EASST	500	r3-200	5	20	5	3	3.34	0.32	256	1.43	31.95	9.58	46.62

Table B.2 Continued from previous page

Planner Name	Its	Map	Tmin	Tmax	DW	DS	W	A	Nodes	Risk	Time (s)	Energy [W-hr]	Flight T(s)
EASST	500	r3-200	5	20	5	3	3.52	0.34	223	1.43	32.49	10.90	28.83
EASST	500	r3-200	5	20	5	3	4.42	0.04	249	1.42	31.39	10.57	29.43
EASST	500	r3-200	5	20	5	3	4.30	0.15	230	1.47	30.79	11.23	47.74
EASST	500	r3-200	5	20	5	3	0.57	0.31	230	1.49	29.86	14.08	26.20
EASST	500	r3-200	5	20	5	3	4.39	0.45	242	1.42	30.49	11.14	26.31
EASST	500	r3-200	5	20	5	3	4.61	0.20	216	1.43	28.62	9.48	50.77
EASST	500	r3-200	5	20	5	3	1.66	0.33	256	1.48	30.93	14.61	16.13
EASST	500	r3-200	5	20	5	3	0.69	0.33	266	1.51	30.25	10.60	57.45
EASST	500	r3-200	5	20	5	3	3.30	0.28	249	1.44	30.93	11.22	55.17
EASST	500	r3-200	5	20	5	3	4.99	0.20	273	1.41	31.17	9.09	54.32
EASST	500	r3-200	5	20	5	3	4.02	0.17	240	1.43	31.26	10.08	41.75
EASST	500	r3-200	5	20	5	3	0.73	0.34	261	1.49	30.49	12.23	58.17
EASST	500	r3-200	5	20	5	3	3.94	0.25	234	1.50	29.82	13.74	37.19
EASST	500	r3-200	5	20	5	3	2.81	0.06	263	1.49	30.92	11.79	49.87

Table B.2 Continued from previous page

Planner Name	Its	Map	Tmin	Tmax	DW	DS	W	A	Nodes	Risk	Time (s)	Energy [W-hr]	Flight T(s)
EASST	500	r3-200	5	20	5	3	2.96	0.02	277	1.44	31.97	10.44	42.27
EASST	500	r3-200	5	20	5	3	3.70	0.22	205	1.44	30.95	10.41	38.61
EASST	500	r3-200	5	20	5	3	3.27	0.35	236	1.49	29.79	13.04	45.26
EASST	500	r3-200	5	20	5	3	4.92	0.40	267	1.44	33.92	9.38	45.99
iEASST	500	14	5	20	5	3	0.97	0.25	323	1.39	122.67	7.88	44.19
iEASST	500	14	5	20	5	3	3.70	0.23	232	1.42	102.00	9.17	42.70
iEASST	500	14	5	20	5	3	1.10	0.28	291	1.41	117.10	9.20	32.59
iEASST	500	14	5	20	5	3	3.35	0.35	225	1.39	101.67	6.94	43.22
iEASST	500	14	5	20	5	3	2.75	0.10	273	1.39	108.49	7.13	45.20
iEASST	500	14	5	20	5	3	0.79	0.10	324	1.38	126.45	7.70	19.70
iEASST	500	14	5	20	5	3	2.12	0.46	284	1.46	119.18	11.79	66.89
iEASST	500	14	5	20	5	3	1.26	0.22	317	1.44	125.67	10.24	38.88
iEASST	500	14	5	20	5	3	2.22	0.11	266	1.39	107.19	7.29	30.90
iEASST	500	14	5	20	5	3	2.06	0.29	307	1.39	126.68	6.13	33.09

Table B.2 Continued from previous page

Planner Name	Its	Map	Tmin	Tmax	DW	DS	W	A	Nodes	Risk	Time (s)	Energy [W-hr]	Flight T(s)
iEASST	500	14	5	20	5	3	4.14	0.18	189	1.44	88.33	9.00	50.38
iEASST	500	14	5	20	5	3	3.98	0.19	264	1.38	110.72	7.37	23.13
iEASST	500	14	5	20	5	3	2.85	0.08	259	1.40	107.72	7.88	29.24
iEASST	500	14	5	20	5	3	0.29	0.12	345	1.47	126.69	11.54	33.89
iEASST	500	14	5	20	5	3	3.56	0.03	253	1.39	103.91	8.26	26.67
iEASST	500	14	5	20	5	3	4.00	0.44	276	1.44	112.82	8.95	57.37
iEASST	500	14	5	20	5	3	4.26	0.39	219	1.39	97.24	9.18	18.06
iEASST	500	14	5	20	5	3	0.37	0.24	292	1.40	115.20	8.34	38.20
iEASST	500	14	5	20	5	3	2.30	0.19	287	1.39	115.25	8.13	17.34
iEASST	500	14	5	20	5	3	2.11	0.38	275	1.61	111.23	16.35	90.80
iEASST	500	14	5	20	5	3	1.57	0.46	269	1.39	109.53	7.20	33.27
iEASST	500	14	5	20	5	3	3.33	0.10	292	1.39	121.74	7.27	37.70
iEASST	500	14	5	20	5	3	0.89	0.48	279	1.39	113.95	8.28	23.68
iEASST	500	14	5	20	5	3	2.92	0.47	274	1.36	112.68	7.02	32.32

Table B.2 Continued from previous page

Planner Name	Its	Map	Tmin	Tmax	DW	DS	W	A	Nodes	Risk	Time (s)	Energy [W-hr]	Flight T(s)
iEASST	500	14	5	20	5	3	1.75	0.36	325	1.40	121.32	8.82	38.60
iEASST	500	14	5	20	5	3	4.40	0.03	226	1.41	99.66	8.33	31.19
iEASST	500	14	5	20	5	3	2.55	0.03	325	1.42	122.31	8.58	39.87
iEASST	500	14	5	20	5	3	0.96	0.28	282	1.43	113.70	8.24	43.29
iEASST	500	14	5	20	5	3	1.10	0.19	348	1.39	124.98	6.78	39.63
iEASST	500	14	5	20	5	3	0.67	0.46	262	1.43	111.34	10.62	42.59
iEASST	500	14	5	20	5	3	1.34	0.43	351	1.39	124.99	7.32	18.78
iEASST	500	14	5	20	5	3	1.76	0.14	314	1.37	120.53	5.45	39.97
iEASST	500	r1-50	5	20	5	3	3.81	0.31	149	1.34	115.80	4.58	12.37
iEASST	500	r1-50	5	20	5	3	3.22	0.04	166	1.33	126.23	3.38	17.59
iEASST	500	r1-50	5	20	5	3	0.70	0.06	181	1.33	128.63	3.82	16.23
iEASST	500	r1-50	5	20	5	3	3.31	0.06	163	1.34	127.63	4.54	11.57
iEASST	500	r1-50	5	20	5	3	1.14	0.32	177	1.34	128.22	4.54	13.53
iEASST	500	r1-50	5	20	5	3	3.77	0.27	147	1.33	121.75	3.10	14.56

Table B.2 Continued from previous page

Planner Name	Its	Map	Tmin	Tmax	DW	DS	W	A	Nodes	Risk	Time (s)	Energy [W-hr]	Flight T(s)
iEASST	500	r1-50	5	20	5	3	4.35	0.23	99	1.32	80.99	3.50	18.02
iEASST	500	r1-50	5	20	5	3	3.86	0.49	113	1.32	94.85	3.84	13.01
iEASST	500	r1-50	5	20	5	3	3.61	0.27	201	1.32	142.69	3.43	14.76
iEASST	500	r1-50	5	20	5	3	0.49	0.39	180	1.33	128.20	4.04	16.74
iEASST	500	r1-50	5	20	5	3	2.75	0.27	171	1.33	132.32	3.67	12.75
iEASST	500	r1-50	5	20	5	3	0.91	0.48	181	1.33	126.26	4.43	15.87
iEASST	500	r1-50	5	20	5	3	3.54	0.03	136	1.34	112.42	4.43	9.08
iEASST	500	r1-50	5	20	5	3	2.00	0.44	192	1.33	129.90	4.40	11.18
iEASST	500	r1-50	5	20	5	3	0.14	0.28	173	1.33	119.52	3.03	19.00
iEASST	500	r1-50	5	20	5	3	0.26	0.11	159	1.33	124.66	3.03	17.57
iEASST	500	r1-50	5	20	5	3	3.96	0.37	115	1.33	99.73	4.22	13.68
iEASST	500	r1-50	5	20	5	3	0.46	0.32	148	1.33	114.78	3.85	23.36
iEASST	500	r1-50	5	20	5	3	1.48	0.08	153	1.34	113.25	4.60	12.99
iEASST	500	r1-50	5	20	5	3	4.34	0.24	103	1.33	92.66	4.04	14.31

Table B.2 Continued from previous page

Planner Name	Its	Map	Tmin	Tmax	DW	DS	W	A	Nodes	Risk	Time (s)	Energy [W-hr]	Flight T(s)
iEASST	500	r1-50	5	20	5	3	1.54	0.38	159	1.34	115.67	4.24	12.97
iEASST	500	r1-50	5	20	5	3	2.39	0.06	150	1.33	97.12	3.66	13.74
iEASST	500	r1-50	5	20	5	3	4.55	0.42	94	1.34	62.94	5.20	16.72
iEASST	500	r1-50	5	20	5	3	2.18	0.29	175	1.34	89.43	4.15	13.83
iEASST	500	r1-50	5	20	5	3	4.70	0.04	87	1.33	64.84	3.26	13.63
iEASST	500	r1-50	5	20	5	3	2.68	0.29	101	1.32	69.85	3.03	14.69
iEASST	500	r1-50	5	20	5	3	0.86	0.01	123	1.34	73.61	4.18	13.69
iEASST	500	r1-50	5	20	5	3	2.09	0.19	192	1.34	106.68	4.33	15.01
iEASST	500	r1-50	5	20	5	3	0.67	0.27	163	1.35	99.98	5.64	22.13
iEASST	500	r1-50	5	20	5	3	3.76	0.25	138	1.33	89.75	3.90	18.42
iEASST	500	r1-50	5	20	5	3	1.67	0.09	149	1.34	74.61	3.89	18.14
iEASST	500	r1-50	5	20	5	3	4.29	0.09	132	1.33	74.86	3.08	14.10
iEASST	500	r2-100	5	20	5	3	3.79	0.25	296	1.39	178.74	9.32	12.03
iEASST	500	r2-100	5	20	5	3	0.96	0.43	352	1.39	194.27	7.21	28.37

Table B.2 Continued from previous page

Planner Name	Its	Map	Tmin	Tmax	DW	DS	W	A	Nodes	Risk	Time (s)	Energy [W-hr]	Flight T(s)
iEASST	500	r2-100	5	20	5	3	1.75	0.15	363	1.40	167.33	7.29	47.09
iEASST	500	r2-100	5	20	5	3	1.97	0.21	358	1.39	172.49	7.84	34.41
iEASST	500	r2-100	5	20	5	3	0.81	0.17	386	1.37	185.30	6.66	34.73
iEASST	500	r2-100	5	20	5	3	4.45	0.27	267	1.37	172.54	6.13	38.15
iEASST	500	r2-100	5	20	5	3	3.16	0.18	290	1.38	148.61	7.93	24.67
iEASST	500	r2-100	5	20	5	3	0.47	0.29	372	1.41	177.84	8.58	43.30
iEASST	500	r2-100	5	20	5	3	2.56	0.37	349	1.40	167.50	8.58	35.11
iEASST	500	r2-100	5	20	5	3	4.57	0.20	259	1.38	157.38	6.28	32.00
iEASST	500	r2-100	5	20	5	3	0.81	0.18	352	1.37	168.46	6.54	26.25
iEASST	500	r2-100	5	20	5	3	3.62	0.41	281	1.39	164.64	7.06	33.55
iEASST	500	r2-100	5	20	5	3	3.53	0.02	282	1.35	155.07	6.08	22.86
iEASST	500	r2-100	5	20	5	3	4.09	0.21	279	1.37	131.90	6.56	36.57
iEASST	500	r2-100	5	20	5	3	4.97	0.47	233	1.37	123.72	7.68	18.53
iEASST	500	r2-100	5	20	5	3	1.35	0.21	289	1.36	123.81	5.75	29.38

Table B.2 Continued from previous page

Planner Name	Its	Map	Tmin	Tmax	DW	DS	W	A	Nodes	Risk	Time (s)	Energy [W-hr]	Flight T(s)
iEASST	500	r2-100	5	20	5	3	2.58	0.40	309	1.40	167.22	9.45	24.23
iEASST	500	r2-100	5	20	5	3	0.60	0.41	382	1.42	205.63	11.78	17.73
iEASST	500	r2-100	5	20	5	3	2.12	0.22	368	1.36	182.26	6.16	27.64
iEASST	500	r2-100	5	20	5	3	4.16	0.22	263	1.38	128.57	7.15	28.34
iEASST	500	r2-100	5	20	5	3	3.00	0.28	305	1.37	154.55	7.44	20.72
iEASST	500	r2-100	5	20	5	3	2.62	0.19	348	1.39	154.94	7.82	38.04
iEASST	500	r2-100	5	20	5	3	2.49	0.07	299	1.40	133.17	8.20	24.71
iEASST	500	r2-100	5	20	5	3	0.03	0.19	395	1.36	145.51	5.00	35.03
iEASST	500	r2-100	5	20	5	3	1.23	0.25	326	1.40	140.70	8.15	45.10
iEASST	500	r2-100	5	20	5	3	4.20	0.05	310	1.41	134.64	9.15	30.28
iEASST	500	r2-100	5	20	5	3	0.21	0.24	346	1.38	144.35	7.19	33.68
iEASST	500	r2-100	5	20	5	3	0.81	0.46	351	1.40	145.01	7.86	32.70
iEASST	500	r2-100	5	20	5	3	2.30	0.18	364	1.42	145.91	9.57	21.74
iEASST	500	r2-100	5	20	5	3	3.11	0.38	321	1.38	139.85	7.11	32.23

Table B.2 Continued from previous page

Planner Name	Its	Map	Tmin	Tmax	DW	DS	W	A	Nodes	Risk	Time (s)	Energy [W-hr]	Flight T(s)
iEASST	500	r2-100	5	20	5	3	3.83	0.02	320	1.38	138.22	7.28	25.67
iEASST	500	r2-100	5	20	5	3	3.87	0.39	320	1.38	137.36	6.70	29.96
iEASST	500	r2-100	5	20	5	3	3.64	0.45	324	1.39	138.96	6.20	34.66
iEASST	500	r2-100	5	20	5	3	4.07	0.38	286	1.38	126.80	8.89	17.36
iEASST	500	r3-200	5	20	5	3	3.36	0.41	415	1.43	166.56	10.01	53.18
iEASST	500	r3-200	5	20	5	3	2.35	0.12	402	1.40	161.06	9.21	36.66
iEASST	500	r3-200	5	20	5	3	4.63	0.49	310	1.40	154.90	8.50	45.68
iEASST	500	r3-200	5	20	5	3	2.66	0.17	395	1.51	169.37	13.50	47.92
iEASST	500	r3-200	5	20	5	3	1.73	0.48	438	1.42	166.64	9.12	43.55
iEASST	500	r3-200	5	20	5	3	4.94	0.04	305	1.41	152.77	8.98	42.96
iEASST	500	r3-200	5	20	5	3	4.34	0.40	353	1.42	163.13	9.83	45.50
iEASST	500	r3-200	5	20	5	3	0.81	0.24	415	1.41	170.02	8.80	54.28
iEASST	500	r3-200	5	20	5	3	3.15	0.04	346	1.46	154.93	11.48	42.11
iEASST	500	r3-200	5	20	5	3	1.71	0.15	396	1.45	163.57	10.09	39.90

Table B.2 Continued from previous page

Planner Name	Its	Map	Tmin	Tmax	DW	DS	W	A	Nodes	Risk	Time (s)	Energy [W-hr]	Flight T(s)
iEASST	500	r3-200	5	20	5	3	4.23	0.06	379	1.41	166.26	9.44	33.42
iEASST	500	r3-200	5	20	5	3	3.09	0.13	387	1.41	162.44	9.21	44.60
iEASST	500	r3-200	5	20	5	3	3.89	0.34	345	1.42	156.57	8.83	51.51
iEASST	500	r3-200	5	20	5	3	3.34	0.32	402	1.50	170.75	13.58	33.47
iEASST	500	r3-200	5	20	5	3	3.52	0.34	372	1.42	158.46	9.64	51.94
iEASST	500	r3-200	5	20	5	3	4.42	0.04	348	1.43	162.42	9.45	40.85
iEASST	500	r3-200	5	20	5	3	4.30	0.15	328	1.47	155.71	13.08	30.97
iEASST	500	r3-200	5	20	5	3	0.57	0.31	399	1.46	165.17	11.83	44.08
iEASST	500	r3-200	5	20	5	3	4.39	0.45	356	1.43	160.05	8.51	55.43
iEASST	500	r3-200	5	20	5	3	4.61	0.20	337	1.40	157.99	8.60	41.33
iEASST	500	r3-200	5	20	5	3	1.66	0.33	419	1.49	168.69	12.91	46.84
iEASST	500	r3-200	5	20	5	3	0.69	0.33	437	1.41	166.41	9.82	50.36
iEASST	500	r3-200	5	20	5	3	3.30	0.28	385	1.46	156.69	12.77	33.89
iEASST	500	r3-200	5	20	5	3	4.99	0.20	275	1.42	142.90	10.19	33.85

Table B.2 Continued from previous page

Planner Name	Its	Map	Tmin	Tmax	DW	DS	W	A	Nodes	Risk	Time (s)	Energy [W-hr]	Flight T(s)
iEASST	500	r3-200	5	20	5	3	4.02	0.17	363	1.42	160.66	10.33	42.91
iEASST	500	r3-200	5	20	5	3	0.73	0.34	417	1.44	166.80	10.66	40.50
iEASST	500	r3-200	5	20	5	3	3.94	0.25	376	1.42	156.58	9.00	48.48
iEASST	500	r3-200	5	20	5	3	2.81	0.06	350	1.43	158.86	11.08	26.23
iEASST	500	r3-200	5	20	5	3	2.96	0.02	384	1.49	160.87	11.66	52.32
iEASST	500	r3-200	5	20	5	3	3.70	0.22	355	1.48	160.11	11.51	41.52
iEASST	500	r3-200	5	20	5	3	3.27	0.35	376	1.45	161.44	9.65	46.44
iEASST	500	r3-200	5	20	5	3	4.92	0.40	322	1.41	158.41	8.74	42.87
SST	500	14	5	20	5	3	0.97	0.25	137	278.46	20.97	9.58	39.63
SST	500	14	5	20	5	3	3.70	0.23	119	233.88	20.26	7.75	26.51
SST	500	14	5	20	5	3	1.10	0.28	152	203.99	20.34	7.88	33.42
SST	500	14	5	20	5	3	3.35	0.35	134	245.50	21.00	8.62	19.05
SST	500	14	5	20	5	3	2.75	0.10	93	203.00	20.98	6.21	34.03
SST	500	14	5	20	5	3	0.79	0.10	129	322.85	21.53	12.25	71.34

Table B.2 Continued from previous page

Planner Name	Its	Map	Tmin	Tmax	DW	DS	W	A	Nodes	Risk	Time (s)	Energy [W-hr]	Flight T(s)
SST	500	14	5	20	5	3	2.12	0.46	118	279.91	21.71	13.02	65.03
SST	500	14	5	20	5	3	1.26	0.22	152	356.91	21.74	15.23	62.60
SST	500	14	5	20	5	3	2.22	0.11	158	245.25	21.92	8.60	60.74
SST	500	14	5	20	5	3	2.06	0.29	127	292.65	21.32	12.15	59.32
SST	500	14	5	20	5	3	4.14	0.18	133	247.34	20.74	8.99	24.89
SST	500	14	5	20	5	3	3.98	0.19	130	233.57	20.10	8.56	38.62
SST	500	14	5	20	5	3	2.85	0.08	141	269.55	20.71	8.61	44.95
SST	500	14	5	20	5	3	0.29	0.12	132	204.67	20.87	6.41	46.49
SST	500	14	5	20	5	3	3.56	0.03	122	208.69	20.52	5.52	35.74
SST	500	14	5	20	5	3	4.00	0.44	94	253.81	21.38	11.06	49.16
SST	500	14	5	20	5	3	4.26	0.39	86	301.60	20.91	12.22	69.70
SST	500	14	5	20	5	3	0.37	0.24	132	250.66	20.54	9.76	23.59
SST	500	14	5	20	5	3	2.30	0.19	156	160.62	20.93	7.58	49.82
SST	500	14	5	20	5	3	2.11	0.38	148	271.98	20.80	10.60	45.10

Table B.2 Continued from previous page

Planner Name	Its	Map	Tmin	Tmax	DW	DS	W	A	Nodes	Risk	Time (s)	Energy [W-hr]	Flight T(s)
SST	500	14	5	20	5	3	1.57	0.46	130	283.08	20.72	9.73	42.80
SST	500	14	5	20	5	3	3.33	0.10	140	245.21	20.61	12.03	75.40
SST	500	14	5	20	5	3	0.89	0.48	141	245.79	21.61	10.90	49.18
SST	500	14	5	20	5	3	2.92	0.47	144	303.06	20.99	15.48	104.62
SST	500	14	5	20	5	3	1.75	0.36	155	200.87	20.85	7.11	28.62
SST	500	14	5	20	5	3	4.40	0.03	109	282.21	20.40	8.76	62.39
SST	500	14	5	20	5	3	2.55	0.03	152	303.15	21.06	11.61	78.48
SST	500	14	5	20	5	3	0.96	0.28	141	335.19	21.13	15.04	77.33
SST	500	14	5	20	5	3	1.10	0.19	156	272.49	20.18	10.15	21.76
SST	500	14	5	20	5	3	0.67	0.46	149	247.45	20.91	8.34	57.74
SST	500	14	5	20	5	3	1.34	0.43	149	257.47	20.68	9.56	50.59
SST	500	14	5	20	5	3	1.76	0.14	128	257.30	20.14	11.81	66.68
SST	500	r1-50	5	20	5	3	3.81	0.31	86	113.70	30.65	3.82	19.26
SST	500	r1-50	5	20	5	3	3.22	0.04	65	114.03	30.99	5.52	27.16

Table B.2 Continued from previous page

Planner Name	Its	Map	Tmin	Tmax	DW	DS	W	A	Nodes	Risk	Time (s)	Energy [W-hr]	Flight T(s)
SST	500	r1-50	5	20	5	3	0.70	0.06	102	99.04	31.16	4.07	21.82
SST	500	r1-50	5	20	5	3	3.31	0.06	64	96.60	30.30	3.86	19.44
SST	500	r1-50	5	20	5	3	1.14	0.32	93	107.47	31.51	5.82	21.43
SST	500	r1-50	5	20	5	3	3.77	0.27	105	93.53	27.23	4.74	27.23
SST	500	r1-50	5	20	5	3	4.35	0.23	100	119.22	27.39	5.01	29.33
SST	500	r1-50	5	20	5	3	3.86	0.49	84	99.89	30.44	3.81	19.28
SST	500	r1-50	5	20	5	3	3.61	0.27	78	97.40	29.97	4.02	18.90
SST	500	r1-50	5	20	5	3	0.49	0.39	79	102.20	30.26	5.28	30.36
SST	500	r1-50	5	20	5	3	2.75	0.27	86	108.50	32.78	4.07	18.67
SST	500	r1-50	5	20	5	3	0.91	0.48	75	120.62	30.65	5.15	21.81
SST	500	r1-50	5	20	5	3	3.54	0.03	64	91.91	28.10	3.30	17.23
SST	500	r1-50	5	20	5	3	2.00	0.44	79	103.78	27.51	5.34	22.99
SST	500	r1-50	5	20	5	3	0.14	0.28	77	102.62	31.85	6.91	39.93
SST	500	r1-50	5	20	5	3	0.26	0.11	86	113.18	31.46	4.82	18.61

Table B.2 Continued from previous page

Planner Name	Its	Map	Tmin	Tmax	DW	DS	W	A	Nodes	Risk	Time (s)	Energy [W-hr]	Flight T(s)
SST	500	r1-50	5	20	5	3	3.96	0.37	96	103.86	32.83	3.53	19.45
SST	500	r1-50	5	20	5	3	0.46	0.32	102	101.12	31.38	4.77	28.07
SST	500	r1-50	5	20	5	3	1.48	0.08	66	123.46	30.68	4.71	9.53
SST	500	r1-50	5	20	5	3	4.34	0.24	77	101.38	30.11	4.40	22.32
SST	500	r1-50	5	20	5	3	1.54	0.38	69	108.63	22.02	4.53	22.41
SST	500	r1-50	5	20	5	3	2.39	0.06	64	121.49	21.52	5.84	33.75
SST	500	r1-50	5	20	5	3	4.55	0.42	87	94.85	22.46	3.18	18.41
SST	500	r1-50	5	20	5	3	2.18	0.29	92	123.83	22.44	6.56	30.04
SST	500	r1-50	5	20	5	3	4.70	0.04	88	102.29	22.01	3.85	16.72
SST	500	r1-50	5	20	5	3	2.68	0.29	102	112.30	21.58	5.03	23.22
SST	500	r1-50	5	20	5	3	0.86	0.01	92	106.21	24.30	4.69	20.09
SST	500	r1-50	5	20	5	3	2.09	0.19	92	115.51	22.22	4.72	22.07
SST	500	r1-50	5	20	5	3	0.67	0.27	96	110.76	25.63	4.56	24.92
SST	500	r1-50	5	20	5	3	3.76	0.25	63	107.06	33.58	4.64	13.77

Table B.2 Continued from previous page

Planner Name	Its	Map	Tmin	Tmax	DW	DS	W	A	Nodes	Risk	Time (s)	Energy [W-hr]	Flight T(s)
SST	500	r1-50	5	20	5	3	1.67	0.09	95	101.86	20.42	6.07	28.84
SST	500	r1-50	5	20	5	3	4.29	0.09	81	97.63	20.51	4.13	27.56
SST	500	r2-100	5	20	5	3	3.79	0.25	180	230.46	27.72	7.81	31.96
SST	500	r2-100	5	20	5	3	0.96	0.43	188	212.32	29.69	8.18	40.09
SST	500	r2-100	5	20	5	3	1.75	0.15	196	219.64	31.20	7.50	33.18
SST	500	r2-100	5	20	5	3	1.97	0.21	219	246.01	29.69	10.17	63.77
SST	500	r2-100	5	20	5	3	0.81	0.17	214	241.43	32.98	9.62	43.68
SST	500	r2-100	5	20	5	3	4.45	0.27	179	238.18	39.66	8.51	31.12
SST	500	r2-100	5	20	5	3	3.16	0.18	206	207.69	30.34	7.65	31.67
SST	500	r2-100	5	20	5	3	0.47	0.29	210	213.82	30.46	6.90	40.00
SST	500	r2-100	5	20	5	3	2.56	0.37	181	208.24	36.94	8.97	47.02
SST	500	r2-100	5	20	5	3	4.57	0.20	169	201.57	27.15	6.93	42.51
SST	500	r2-100	5	20	5	3	0.81	0.18	193	233.36	32.21	7.77	31.10
SST	500	r2-100	5	20	5	3	3.62	0.41	206	203.72	26.30	6.40	28.38

Table B.2 Continued from previous page

Planner Name	Its	Map	Tmin	Tmax	DW	DS	W	A	Nodes	Risk	Time (s)	Energy [W-hr]	Flight T(s)
SST	500	r2-100	5	20	5	3	3.53	0.02	201	215.72	24.83	7.08	24.45
SST	500	r2-100	5	20	5	3	4.09	0.21	169	194.32	25.88	6.03	25.19
SST	500	r2-100	5	20	5	3	4.97	0.47	162	217.60	24.47	6.18	30.00
SST	500	r2-100	5	20	5	3	1.35	0.21	174	202.60	25.37	7.30	42.42
SST	500	r2-100	5	20	5	3	2.58	0.40	178	235.89	28.11	8.77	25.79
SST	500	r2-100	5	20	5	3	0.60	0.41	214	220.27	27.74	8.96	54.61
SST	500	r2-100	5	20	5	3	2.12	0.22	189	225.50	26.29	7.64	38.89
SST	500	r2-100	5	20	5	3	4.16	0.22	182	232.94	24.71	10.63	56.06
SST	500	r2-100	5	20	5	3	3.00	0.28	186	197.47	25.17	6.54	28.39
SST	500	r2-100	5	20	5	3	2.62	0.19	192	209.79	23.95	6.65	28.57
SST	500	r2-100	5	20	5	3	2.49	0.07	190	245.02	24.01	8.76	43.29
SST	500	r2-100	5	20	5	3	0.03	0.19	202	231.03	24.67	6.57	23.67
SST	500	r2-100	5	20	5	3	1.23	0.25	199	227.78	24.21	8.87	34.88
SST	500	r2-100	5	20	5	3	4.20	0.05	185	214.39	23.98	7.16	33.42

Table B.2 Continued from previous page

Planner Name	Its	Map	Tmin	Tmax	DW	DS	W	A	Nodes	Risk	Time (s)	Energy [W-hr]	Flight T(s)
SST	500	r2-100	5	20	5	3	0.21	0.24	193	213.60	23.86	4.66	30.95
SST	500	r2-100	5	20	5	3	0.81	0.46	203	233.92	24.40	8.61	35.12
SST	500	r2-100	5	20	5	3	2.30	0.18	182	210.77	23.81	7.14	38.53
SST	500	r2-100	5	20	5	3	3.11	0.38	180	218.16	23.64	7.28	23.18
SST	500	r2-100	5	20	5	3	3.83	0.02	179	219.56	24.30	8.67	52.29
SST	500	r2-100	5	20	5	3	3.87	0.39	186	243.03	24.19	8.30	40.49
SST	500	r2-100	5	20	5	3	3.64	0.45	176	206.51	24.11	6.65	35.77
SST	500	r2-100	5	20	5	3	4.07	0.38	159	221.26	22.89	7.28	32.44
SST	500	r3-200	5	20	5	3	3.36	0.41	257	361.84	31.71	10.89	39.24
SST	500	r3-200	5	20	5	3	2.35	0.12	290	348.27	33.82	11.91	49.85
SST	500	r3-200	5	20	5	3	4.63	0.49	245	333.85	36.75	11.57	31.41
SST	500	r3-200	5	20	5	3	2.66	0.17	255	329.23	30.11	9.90	57.75
SST	500	r3-200	5	20	5	3	1.73	0.48	275	315.60	31.34	9.64	49.19
SST	500	r3-200	5	20	5	3	4.94	0.04	208	329.28	30.10	8.15	53.68

Table B.2 Continued from previous page

Planner Name	Its	Map	Tmin	Tmax	DW	DS	W	A	Nodes	Risk	Time (s)	Energy [W-hr]	Flight T(s)
SST	500	r3-200	5	20	5	3	4.34	0.40	231	337.75	30.88	12.51	73.92
SST	500	r3-200	5	20	5	3	0.81	0.24	271	369.75	29.65	13.94	24.70
SST	500	r3-200	5	20	5	3	3.15	0.04	260	304.46	31.12	9.45	51.92
SST	500	r3-200	5	20	5	3	1.71	0.15	252	361.72	30.01	14.29	73.67
SST	500	r3-200	5	20	5	3	4.23	0.06	262	335.43	30.08	10.69	25.21
SST	500	r3-200	5	20	5	3	3.09	0.13	258	358.09	30.64	12.23	33.90
SST	500	r3-200	5	20	5	3	3.89	0.34	235	327.66	29.89	9.72	51.33
SST	500	r3-200	5	20	5	3	3.34	0.32	259	344.53	31.34	16.48	92.59
SST	500	r3-200	5	20	5	3	3.52	0.34	247	346.92	30.11	14.90	63.13
SST	500	r3-200	5	20	5	3	4.42	0.04	251	338.27	30.85	9.16	51.67
SST	500	r3-200	5	20	5	3	4.30	0.15	250	323.64	29.78	9.70	45.23
SST	500	r3-200	5	20	5	3	0.57	0.31	306	377.03	31.44	10.75	59.82
SST	500	r3-200	5	20	5	3	4.39	0.45	248	314.52	29.28	10.52	46.28
SST	500	r3-200	5	20	5	3	4.61	0.20	279	326.52	31.83	10.06	45.22

Table B.2 Continued from previous page

Planner Name	Its	Map	Tmin	Tmax	DW	DS	W	A	Nodes	Risk	Time (s)	Energy [W-hr]	Flight T(s)
SST	500	r3-200	5	20	5	3	1.66	0.33	268	346.94	29.64	11.38	46.65
SST	500	r3-200	5	20	5	3	0.69	0.33	267	392.52	30.37	12.92	55.64
SST	500	r3-200	5	20	5	3	3.30	0.28	249	346.28	29.62	12.22	62.90
SST	500	r3-200	5	20	5	3	4.99	0.20	189	321.13	29.89	8.91	48.59
SST	500	r3-200	5	20	5	3	4.02	0.17	269	319.35	30.57	9.38	32.64
SST	500	r3-200	5	20	5	3	0.73	0.34	274	317.32	29.96	10.40	56.14
SST	500	r3-200	5	20	5	3	3.94	0.25	222	346.94	30.17	12.62	60.76
SST	500	r3-200	5	20	5	3	2.81	0.06	285	347.56	30.79	10.36	46.09
SST	500	r3-200	5	20	5	3	2.96	0.02	287	341.77	30.33	12.15	43.03
SST	500	r3-200	5	20	5	3	3.70	0.22	263	352.57	31.36	15.10	98.18
SST	500	r3-200	5	20	5	3	3.27	0.35	182	340.28	26.46	9.30	39.66
SST	500	r3-200	5	20	5	3	4.92	0.40	251	317.46	30.04	14.36	91.97

B.2 Pix2Pix Supplemental Figures

This section provides additional figures that were not included in the core of Chapter 3. The Pix2Pix generator and discriminator as described by the TensorFlow package is shown in Figure B.1. The generator on the left is a flowchart of the encoder-decoder network, U-Net, showing how it uses skip connections from previous layers. The input is a 256x256x5 tensor, for the five input channels of the training data, and the output is a 256x256x3 tensor, for the three output channels that represent the class labels in the RGB encoding. The right is the discriminator network that takes as input the generated image and target image, for real training data samples. The output is a 30x30 prediction of real or fake data and is compared to an array of zeros or ones depending on if the data is fake or not.

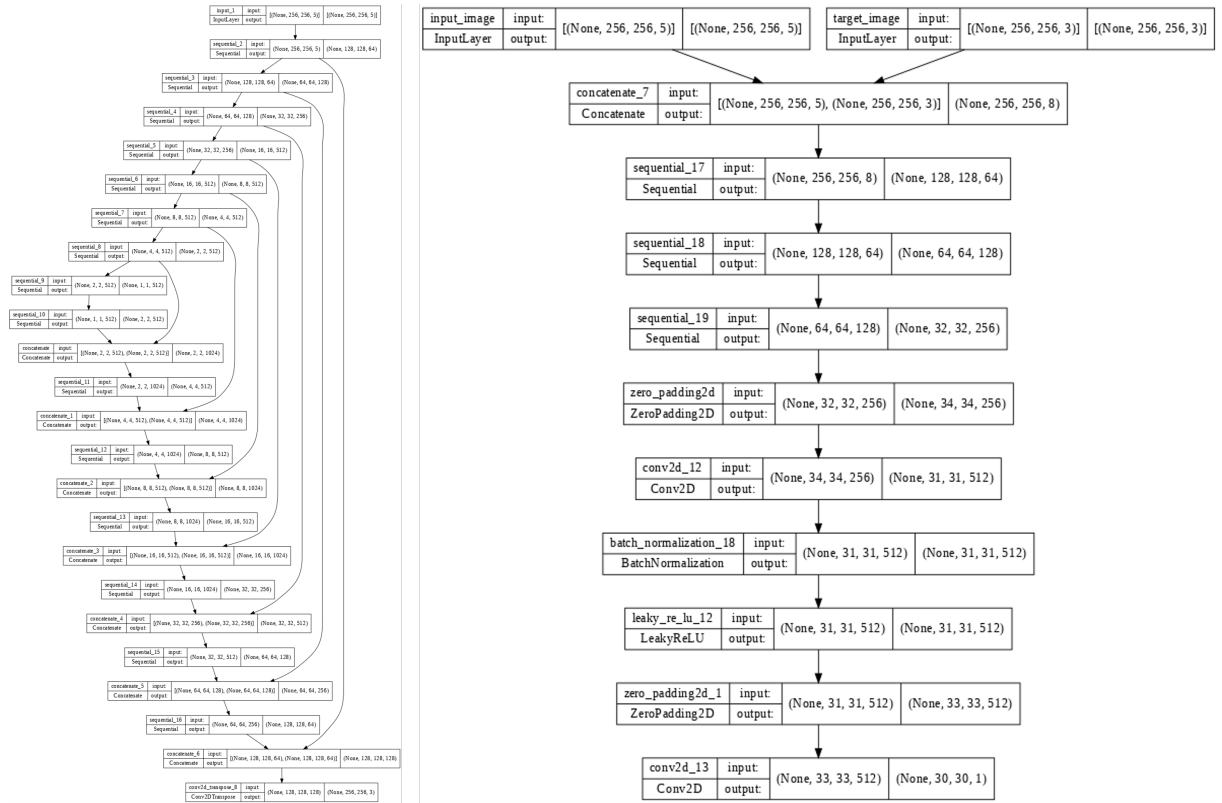


Figure B.1: Generator and Discriminator Models Respectively, Produced by Tensorflow

Figure B.2 visualizes the results from the 5-channel one-hot encoding that was used

to compare against the 3-channel RGB encoding results. The display is provided here to compare the results qualitatively to the RGB encoding results. For instance, no unknown labels are seen and the structure predictions are not as precise nor do they have as clean edges. For this model, the default appears to become the parks label as opposed to the unknown label. This may have to do with the order of the labels in the one-hot encoding, or more generally, because of the use of a unique class for the unknown label as opposed to a prediction threshold.

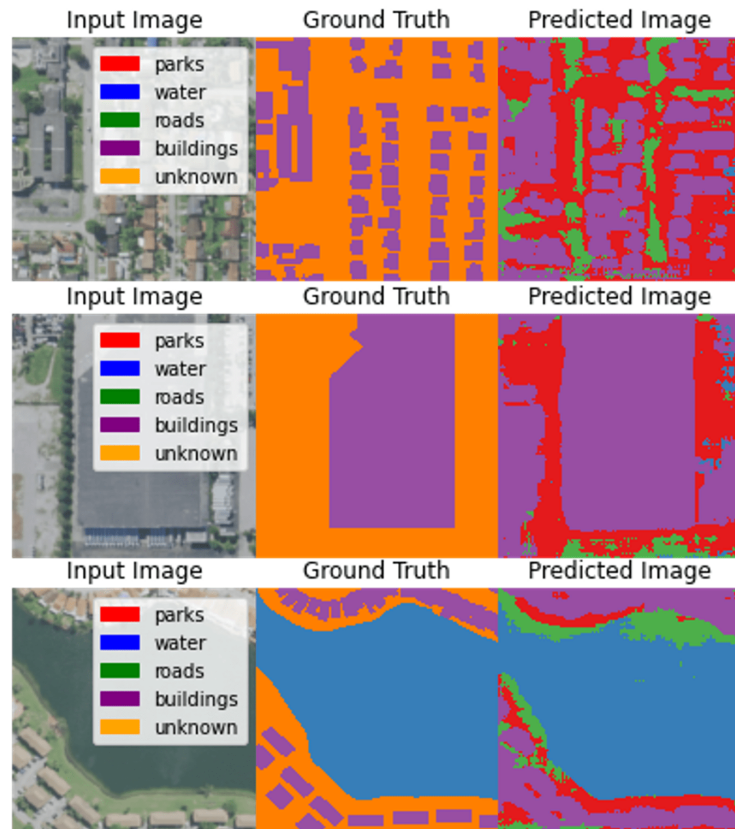


Figure B.2: Training Results for 5-Channel One-hot Encoding

The Pix2Pix predictions were originally hypothesized to improve upon current benchmarks from USGS Landcover data and Google Dynamic World predictions. A mapping between the labels is shown in Figure B.3 to help in matching the visual results shown in Chapter 3. In addition, the contrast of the quantity of classes and detail of the class types is

clear between the benchmark data sources and the RUM's Pix2Pix predictions.

NLCD	Dynamic World	RUM Training	RUM Testing
Open Water	Water	Water	Water
Ice/Snow	Snow and Ice	Background	Background
Developed, OS	Built-up Area	Roads Parking Lots Sidewalks	Concrete
Developed, LI		Structures	Structures
Developed, MI			
Developed, HI			
Rock/Sand/Clay	Bare Ground	Background	Background
Dec Forest	Trees	Trees	Greenspace
Ever Forest			
Mixed Forest			
Dwarf Shrub	Shrub	Greenspace	
Shrub			
Grassland	Grass		
Sedge			
Lichens			
Moss			
Pasture	Crops	Background	Background
Crops			
Wetlands	Flooded Vegetation	Background	
EH Wetlands			
...	...	Pole/Other	Pole/Other

Figure B.3: Mapping Between Labels of NLCD and Dynamic World to Custom RUM Labeling

APPENDIX C

KEY SOFTWARE RESOURCES

C.1 Python Tools

A special thanks to the developers and contributors of the following python packages. The packages are used throughout the SAFER software framework and are required for the final demonstration experiment.

- Geopandas
- Rasterio
- Numpy
- SciPy
- NURBS-Python (<https://github.com/orbingol/NURBS-Python>)
- cvar-energy-risk-deep-model (<https://github.com/castacks/cvar-energy-risk-deep-model>)
- Matplotlib
- Plotly

The Google cloud ecosystem and infrastructure were critical to the geospatial data processing and deep learning sections of this thesis. Google Earth Engine and Google Colab helped complete the project, while using a single laptop, the Lenovo Thinkpad P51.

C.2 Data and Software Access

If you would like access to the data or software, please email the author at caleb.harris94@gatech.edu, or contact him through ResearchGate or LinkedIn.

REFERENCES

- [1] D. Merrick, “World disasters report 2020: Come heat or high water - tackling the humanitarian impacts of the climate crisis together,” International Federation of Red Cross and Red Crescent Societies, Report.
- [2] R. R. Murphy, *Disaster Robotics*. MIT Press, 2014, ISBN: 9780262321303.
- [3] D. Helbing, H. Ammoser, and C. Kühnert, “Disasters as extreme events and the importance of network interactions for disaster response management,” in *Extreme Events in Nature and Society*, ser. The Frontiers Collection. 2006, ch. 15, pp. 319–348.
- [4] *Disaster Resilience: A National Imperative*. Washington, DC: The National Academies Press, 2012, ISBN: 978-0-309-26150-0.
- [5] *U.S. Billion-Dollar Weather and Climate Disasters*. NOAA National Centers for Environmental Information (NCEI), 2022.
- [6] “Climate change 2021: The physical science basis,” Intergovernmental Panel on Climate Change, Report, 2021.
- [7] M. Kii, “Projecting future populations of urban agglomerations around the world and through the 21st century,” *Urban Sustainability*, vol. 1, no. 1, 2021.
- [8] United Nations, *68% of the world population projected to live in urban areas by 2050, says un*, Web Page, 2018.
- [9] J. Son, S. Brown, Z. Aziz, and F. Peña-Mora, “Supporting disaster response and recovery through improved situation awareness,” *Structural Survey*, vol. 26, no. 5, pp. 411–425, 2008.
- [10] “Fema unmanned aerial systems program,” Subcommittee on Disaster Reduction, Presentation.
- [11] T. Wachtendorf, “Improvising 9/11 : Organizational improvisation following the world trade center disaster,” Ph.D. dissertation, University of Delaware, 2004.
- [12] A. Dimou *et al.*, “Faster: First responder advanced technologies for safe and efficient emergency response,” in *Technology Development for Security Practitioners*. 2021, pp. 447–460, ISBN: 978-3-030-69459-3.

- [13] J. Li, S. Zlatanova, and A. G. Fabbri, *Geomatics Solutions for Disaster Management*, ser. Lecture Notes in Geoinformation and Cartography. Berlin, Heidelberg: Springer, 2007.
- [14] G. Lewis, "Evaluating the use of a low-cost unmanned aerial vehicle platform in acquiring digital imagery for emergency response," in Jul. 2007, vol. 117, pp. 117–133, ISBN: 978-3-540-72106-2.
- [15] U. N. O. for the Coordination of Humanitarian Efforts, "Unmanned aerial vehicles in humanitarian response," Tech. Rep., 2014.
- [16] V. Kumar, D. Rus, and S. Singh, "Robot and sensor networks for first responders," *Pervasive Computing, IEEE*, vol. 3, pp. 24–33, Nov. 2004.
- [17] N. Schurr, J. Marecki, M. Tambe, P. Scerri, N. Kasinadhuni, and J. Lewis, "The future of disaster response: Humans working with multiagent teams using defacto," Jan. 2005, pp. 9–16.
- [18] V. Kumar and N. Michael, "Opportunities and challenges with autonomous micro aerial vehicles," *The International Journal of Robotics Research*, vol. 31, no. 11, pp. 1279–1291, 2012.
- [19] C. Luo, W. Miao, H. Ullah, S. McClean, G. Parr, and G. Min, "Unmanned aerial vehicles for disaster management," in Aug. 2019, pp. 83–107, ISBN: 978-981-13-0991-5.
- [20] J. Delmerico *et al.*, "The current state and future outlook of rescue robotics," *Journal of Field Robotics*, vol. 36, no. 7, pp. 1171–1191, 2019.
- [21] M. K. Habib and Y. Baudoin, "Robot-assisted risky intervention, search, rescue and environmental surveillance," *International Journal of Advanced Robotic Systems*, vol. 7, no. 1, p. 10, 2010.
- [22] H. Hildmann and E. Kovacs, "Review: Using unmanned aerial vehicles as mobile sensing platforms for disaster response, civil security and public safety," *Drones*, vol. 3, no. 3, 2019.
- [23] R. Murphy, S. Tadokoro, and A. Kleiner, "Disaster robotics," in *Springer Handbook of Robotics*. Springer International Publishing, Jan. 2016, pp. 1577–1604, Publisher Copyright: © Springer-Verlag Berlin Heidelberg 2016., ISBN: 9783319325507.
- [24] V. Jorge *et al.*, "A survey on unmanned surface vehicles for disaster robotics: Main challenges and directions," *Sensors (Basel)*, vol. 19, no. 3, 2019.

- [25] F. Greenwood, E. L. Nelson, and P. G. Greenough, “Flying into the hurricane: A case study of UAV use in damage assessment during the 2017 hurricanes in texas and florida,” *PLoS One*, vol. 15, no. 2, 2020.
- [26] M. Zhou, H. Dong, S. Ge, X. Wang, and F.-Y. Wang, “Robot-guided crowd evacuation in a railway hub station in case of emergencies,” *Journal of Intelligent & Robotic Systems*, vol. 104, Apr. 2022.
- [27] D. Price, “Unmanned aircraft systems for emergency management: A guide for policy makers and practitioners,” Ph.D. dissertation, Naval Postgraduate School, 2016.
- [28] M. Erdelj, E. Natalizio, K. Chowdhury, and I. Akyildiz, “Help from the sky: Leveraging UAVs for disaster management,” *IEEE Pervasive Computing*, vol. 16, pp. 24–32, Jan. 2017.
- [29] J. F. Gilmore, “A knowledge-based autonomous vehicle system for emergency management support,” in *Ryan K., Sutcliffe R.F.E. (eds) AI and Cognitive Science ’92. Workshops in Computing*, 1993.
- [30] T. S. Durrani, W. Wang, and S. M. Forbes, *Geological Disaster Monitoring Based on Sensor Networks*. Springer, 2019, ISBN: 978-981-13-0991-5.
- [31] B. Lindqvist, C. Kanellakis, S. Mansouri, A. Agha-mohammadi, and G. Nikolakopoulos, “Compra: A compact reactive autonomy framework for subterranean mav based search-and-rescue operations,” Aug. 2021.
- [32] I. Colomina and P. Molina, “Unmanned aerial systems for photogrammetry and remote sensing: A review,” *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 92, pp. 79–97, Jun. 2014.
- [33] S. Tang and V. Kumar, “Autonomous flight,” *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 1, no. 1, pp. 29–52, 2018.
- [34] Y. Bestaoui, *Planning and Decision Making for Aerial Robots*. Feb. 2014, vol. 71, ISBN: ISBN 978-3-319-03706-6.
- [35] S. Scherer, “Low-altitude operation of unmanned rotorcraft,” PhD thesis, Carnegie Mellon University, Pittsburgh, PA, USA, 2011.
- [36] Y. Choi, H. Jimenez, and D. Mavris, “Two-layer obstacle collision avoidance with machine learning for more energy-efficient unmanned aircraft trajectories,” *Robotics and Autonomous Systems*, vol. 98, 2017.

- [37] P. Donato, “Toward autonomous aircraft emergency landing planning,” Ph.D. dissertation, University of Michigan, Jan. 2017.
- [38] M. Pfeiffer *et al.*, “Reinforced imitation: Sample efficient deep reinforcement learning for mapless navigation by leveraging prior demonstrations,” *IEEE Robotics and Automation Letters* Volume 3, 2018.
- [39] S. M. LaValle, *Planning Algorithms*. Cambridge University Press, 2006, ISBN: 978-0521862059.
- [40] K. Sedighi, K. Ashenayi, T. Manikas, R. Wainwright, and H.-M. Tai, “Autonomous local path planning for a mobile robot using a genetic algorithm,” vol. 2, Jul. 2004, 1338–1345 Vol.2, ISBN: 0-7803-8515-2.
- [41] K. Kang, “Online optimal obstacle avoidance for rotary-wing autonomous unmanned aerial vehicles,” Ph.D. dissertation, Georgia Institute of Technology, 2012.
- [42] Y. Choi, “A framework for modeling and simulation of control, navigation, and surveillance for unmanned aircraft separation assurance,” Ph.D. dissertation, Georgia Institute of Technology, Atlanta GA USA, Aug. 2016.
- [43] J. Mattingley, Y. Wang, and S. Boyd, “Receding horizon control,” *IEEE Control Systems Magazine*, vol. 31, no. 3, pp. 52–65, 2011.
- [44] D. Malyuta *et al.*, *Convex optimization for trajectory generation*, 2021.
- [45] L. Blackmore, B. Açikmeşe, and J. M. Carson, “Lossless convexification of control constraints for a class of nonlinear optimal control problems,” *Systems & Control Letters*, vol. 61, no. 8, pp. 863–870, 2012.
- [46] S. Kim, C. M. Harris, C. Y. Justin, and D. N. Mavris, “Optimal trajectory and en-route contingency planning for urban air mobility considering battery energy levels,” in *AIAA AVIATION 2022 Forum*.
- [47] S. Karaman and E. Frazzoli, “Sampling-based algorithms for optimal motion planning,” *The International Journal of Robotics Research*, vol. 30, no. 7, pp. 846–894, 2011.
- [48] C. M. Harris, S. Kim, A. P. Payan, and D. N. Mavris, “Emergency planning for aerial vehicles by approximating risk with aerial imagery and geographic data,” in *AIAA SCITECH 2022 Forum*.
- [49] C. Ochoa, “Metalevel motion planning for unmanned aircraft systems: Metrics definition and algorithm selection,” Ph.D. dissertation, University of Michigan, 2021.

- [50] K. Hauser and Y. Zhou, "Asymptotically optimal planning by feasible kinodynamic planning in state-cost space," *IEEE Transactions on Robotics*, vol. PP, May 2015.
- [51] E. M. Atkins, I. A. Portillo, and M. J. Strube, "Emergency flight planning applied to total loss of thrust," *Journal of Aircraft*, vol. 43, no. 4, pp. 1205–1216, 2006.
- [52] Y. Li, Z. Littlefield, and K. Bekris, "Asymptotically optimal sampling-based kinodynamic planning," *The International Journal of Robotics Research*, vol. 35, Jul. 2014.
- [53] X. Hu, B. Pang, F. Dai, and K. H. Low, "Risk assessment model for UAV cost-effective path planning in urban environments," *IEEE Access*, vol. 8, pp. 150 162–150 173, 2020.
- [54] S. Primatesta, L. S. Cuomo, G. Guglieri, and A. Rizzo, "An innovative algorithm to estimate risk optimum path for unmanned aerial vehicles in urban environments," *Transportation Research Procedia*, vol. 35, pp. 44–53, 2018.
- [55] S. Primatesta, A. Rizzo, and A. la Cour-Harbo, "Ground risk map for unmanned aircraft in urban environments," *Journal of Intelligent & Robotic Systems*, vol. 97, no. 3-4, pp. 489–509, 2019.
- [56] S. Primatesta, G. Guglieri, and A. Rizzo, "A risk-aware path planning strategy for UAVs in urban environments," *Journal of Intelligent & Robotic Systems*, vol. 95, no. 2, pp. 629–643, 2018.
- [57] J. Ding, C. J. Tomlin, L. R. Hook, and J. Fuller, "Initial designs for an automatic forced landing system for safer inclusion of small unmanned air vehicles into the national airspace," in *2016 IEEE/AIAA 35th Digital Avionics Systems Conference (DASC)*, 2016.
- [58] X. Hu, B. Pang, F. Dai, and K. H. Low, "Risk assessment model for UAV cost-effective path planning in urban environments," *IEEE Access*, Aug. 2020.
- [59] P. F. Di Donato and E. M. Atkins, "Evaluating risk to people and property for aircraft emergency landing planning," *Journal of Aerospace Information Systems*, vol. 14, no. 5, pp. 259–278, 2017.
- [60] A. J. Ten Harmsel, I. J. Olson, and E. M. Atkins, "Emergency flight planning for an energy-constrained multicopter," *Journal of Intelligent & Robotic Systems*, vol. 85, no. 1, pp. 145–165, 2017.
- [61] J. Slama, P. Vana, and J. Faigl, "Risk-aware trajectory planning in urban environments with safe emergency landing guarantee," in *2021 IEEE 17th International Conference on Automation Science and Engineering (CASE)*, pp. 1606–1612.

- [62] X. Xiao, “Risk-aware path and motion planning for a tethered aerial visual assistant in unstructured or confined environments,” Ph.D. dissertation, Texas A&M University, 2019.
- [63] D. Fan, “Safe robot planning and control using uncertainty-aware deep learning,” Ph.D. dissertation, Georgia Institute of Technology, Atlanta GA USA, Aug. 2021.
- [64] S. Choudhury, “Adaptive motion planning,” Ph.D. dissertation, Carnegie Mellon University, Pittsburgh, PA, USA, Feb. 2018.
- [65] C. A. Ochoa and E. M. Atkins, “Urban metric maps for small unmanned aircraft systems motion planning,” *Journal of Aerospace Information Systems*, vol. 0, no. 0, pp. 1–16, 0.
- [66] A. Choudhry, B. Moon, J. Patrikar, C. Samaras, and S. Scherer, “CVaR-based flight energy risk assessment for multirotor UAVs using a deep energy model,” in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, May 2021.
- [67] F. D. S. Barbosa, “Towards safer and risk-aware motion planning and control for robotic systems,” Ph.D. dissertation, KTH Royal Institute of Technology, 2022.
- [68] J. Faigl, “Multi-goal path planning for cooperative sensing,” Ph.D. dissertation, Czech Technical University in Prague, 2010.
- [69] A. Hakobyan, G. C. Kim, and I. Yang, “Risk-aware motion planning and control using CVaR-constrained optimization,” *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 3924–3931, 2019.
- [70] S. Sarykalin, G. Serraino, and S. Uryasev, “Value-at-risk vs. conditional value-at-risk in risk management and optimization,” in *State-of-the-Art Decision-Making Tools in the Information-Intensive Age*, ch. Chapter 13, pp. 270–294.
- [71] R. Thakker *et al.*, “Autonomous off-road navigation over extreme terrains with perceptually-challenging conditions,” Jan. 2021.
- [72] M. Ryll, J. Ware, J. Carter, and N. Roy, “Semantic trajectory planning for long-distant unmanned aerial vehicle navigation in urban environments,” *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1551–1558, 2020.
- [73] J. Kim and E. Atkins, “Airspace geofencing and flight planning for low-altitude, urban, small unmanned aircraft systems,” *Applied Sciences*, vol. 12, no. 2, 2022.

- [74] Y. Choi, “A framework for concurrent design and route planning optimization of unmanned aerial vehicle based urban delivery systems,” Ph.D. dissertation, Georgia Institute of Technology, Atlanta GA USA, May 2019.
- [75] C. M. Harris, A. P. Payan, and D. N. Mavris, “Detection of obstacle-free landing zones in crowded environments for aerial system emergency planning,” in *Vertical Flight Society’s 9th Biennial Autonomous VTOL Technical Meeting*.
- [76] R. Bellman, “Dynamic programming,” *Science*, vol. 153, no. 3731, pp. 34–37, 1966.
- [77] H. Ledoux, K. A. Ohori, and R. Peters, *Computational modelling of terrains*. Nov. 2021.
- [78] K. A. Ohori, H. Ledoux, and R. Peters, *3D modelling of the built environment*. Feb. 2022.
- [79] Y. Choi, Y. Choi, S. Briceno, and D. N. Mavris, “Three-dimensional UAS trajectory optimization for remote sensing in an irregular terrain environment,” in *2018 International Conference on Unmanned Aircraft Systems (ICUAS)*, 2018, pp. 1101–1108.
- [80] Y. Choi, S. Briceno, and D. Mavris, “Multi-UAV trajectory optimization and deep learning-based imagery analysis for a uas-based inventory tracking solution,” Jan. 2019.
- [81] P. Virtanen *et al.*, “SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python,” *Nature Methods*, vol. 17, pp. 261–272, 2020.
- [82] O. R. Bingol and A. Krishnamurthy, “NURBS-Python: An open-source object-oriented NURBS modeling framework in Python,” *SoftwareX*, vol. 9, pp. 85–94, 2019.
- [83] Y. Choi, M. Chen, Y. Choi, S. Briceno, and D. Mavris, “Multi-UAV trajectory optimization utilizing a NURBS-based terrain model for an aerial imaging mission,” *Journal of Intelligent & Robotic Systems*, vol. 97, Jan. 2020.
- [84] S. Vasudevan, F. Ramos, E. Nettleton, H. Durrant-Whyte, and A. Blair, “Gaussian process modeling of large scale terrain,” in *2009 IEEE International Conference on Robotics and Automation*, 2009, pp. 1047–1053.
- [85] S. S. Inc, “Feature manipulation engine (FME) software,” 2020.

- [86] A. Krizhevsky, I. Sutskever, and G. Hinton, “Imagenet classification with deep convolutional neural networks,” *Neural Information Processing Systems*, vol. 25, Jan. 2012.
- [87] O. Ronneberger, P. Fischer, and T. Brox, “U-Net: Convolutional networks for biomedical image segmentation,” in *International Conference on Medical image computing and computer-assisted intervention*, Springer, 2015, pp. 234–241.
- [88] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, *Image-to-image translation with conditional adversarial networks*, 2018. arXiv: 1611.07004.
- [89] S. Chen, X. Wu, M. W. Mueller, and K. Sreenath, “Real-time geo-localization using satellite imagery and topography for unmanned aerial vehicles,” 2021. arXiv: 2108.03344.
- [90] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, <http://www.deeplearningbook.org>.
- [91] P. Wang, C. Huang, and J. C. Tilton, “Mapping three-dimensional urban structure by fusing landsat and global elevation data,” 2018.
- [92] A. Van Etten, “You only look twice: Rapid multi-scale object detection in satellite imagery,” *arXiv:1805.09512*, 2018.
- [93] Y. Liu, S. Piramanayagam, S. T. Monteiro, and E. Saber, “Semantic segmentation of multisensor remote sensing imagery with deep convnets and higher-order conditional random fields,” *Journal of Applied Remote Sensing*, vol. 13, no. 01, 2019.
- [94] Z. Li, J. D. Wegner, and A. Lucchi, “Topological map extraction from overhead images,” in *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 1715–1724.
- [95] M. Rahnemoonfar, T. Chowdhury, A. Sarkar, D. Varshney, M. Yari, and R. Murphy, “Floodnet: A high resolution aerial imagery dataset for post flood scene understanding,” Dec. 2020.
- [96] L. Hashemi-Beni and A. A. Gebrehiwot, “Flood extent mapping: An integrated method using deep learning and region growing using UAV optical data,” *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 14, pp. 2127–2135, 2021.
- [97] M. Mittal, R. Mohan, W. Burgard, and A. Valada, “Vision-based autonomous UAV navigation and landing for urban search and rescue,” in *International Symposium on Robotics Research (ISRR)*, 2019.

- [98] A. Chakrabarty and C. A. Ippolito, “Wildfire monitoring using unmanned aerial vehicles operating under utm (stereo),” in *AIAA Scitech 2021 Forum*.
- [99] R. Szeliski, *Computer vision algorithms and applications*. London; New York: Springer, 2011.
- [100] C. M. Harris, G. Achour, A. P. Payan, and D. N. Mavris, “Use of machine learning to create a database of wires for helicopter wire strike prevention,” in *AIAA Scitech 2021 Forum*.
- [101] B. Neupane, T. Horanont, and J. Aryal, “Deep learning-based semantic segmentation of urban features in satellite images: A review and meta-analysis,” *Remote Sensing*, vol. 13, no. 4, 2021.
- [102] A. Kulkarni, T. Mohandoss, D. Northrup, E. Mwebaze, and H. Alemohammad, “Semantic segmentation of medium-resolution satellite imagery using conditional generative adversarial networks,” 2020.
- [103] H. T. Aung, S. H. Pha, and W. Takeuchi, “Building footprint extraction in yangon city from monocular optical satellite image using deep learning,” *Geocarto International*, vol. 37, no. 3, pp. 792–812, 2022.
- [104] M. Shah, M. Gupta, and P. Thakkar, “Satgan: Satellite image generation using conditional adversarial networks,” in *2021 International Conference on Communication information and Computing Technology (ICCICT)*, 2021, pp. 1–6.
- [105] M. Sameen, B. Pradhan, and O. Aziz, “Classification of very high resolution aerial photos using spectral-spatial convolutional neural networks,” *Journal of Sensors*, May 2018.
- [106] C. Li and M. Wand, “Precomputed real-time texture synthesis with markovian generative adversarial networks,” *CoRR*, vol. abs/1604.04382, 2016. arXiv: 1604.04382.
- [107] L. Mescheder, A. Geiger, and S. Nowozin, “Which training methods for gans do actually converge?,” 2018.
- [108] W. Liu, F. Su, X. Jin, H. Li, and R. Qin, “Bispace domain adaptation network for remotely sensed semantic segmentation,” *IEEE Transactions on Geoscience and Remote Sensing*, vol. 60, pp. 1–11, 2022.
- [109] G. Cinar, “A methodology for dynamic sizing of electric power generation and distribution architectures,” Ph.D. dissertation, Georgia Institute of Technology, 2018.

- [110] H. Lee, C. Harris, A. Payan, and D. Mavris, “Risk-aware trajectory planning using energy-based analysis for aerial vehicles,” in *AIAA Aviation 2021 Forum*, 2021.
- [111] S. Karaman and E. Frazzoli, “Optimal kinodynamic motion planning using incremental sampling-based methods,” in *49th IEEE Conference on Decision and Control (CDC)*, 2010, pp. 7681–7687.
- [112] A. Sharir and A. Baltsan, “On shortest paths amidst convex polyhedra,” in *Proceedings of the Second Annual Symposium on Computational Geometry*, ser. SCG ’86, Yorktown Heights, New York, USA: Association for Computing Machinery, 1986, pp. 193–206, ISBN: 0897911946.
- [113] K. Solovey, L. Janson, E. Schmerling, E. Frazzoli, and M. Pavone, “Revisiting the asymptotic optimality of RRT*,” 2019.
- [114] T. Bera, D. Ghose, and S. Suresh, “Asymptotic optimality of rapidly exploring random tree,” 2017.
- [115] I. Noreen, A. Khan, and Z. Habib, “Optimal path planning using RRT* based approaches: A survey and future directions,” *International Journal of Advanced Computer Science and Applications*, vol. 7, Nov. 2016.
- [116] S. S. Joshi, S. Hutchinson, and P. Tsiotras, “Tie: Time-informed exploration for robot motion planning,” *IEEE Robotics and Automation Letters*, vol. 6, no. 2,
- [117] O. Arslan and P. Tsiotras, “Use of relaxation methods in sampling-based algorithms for optimal motion planning,” in *2013 IEEE International Conference on Robotics and Automation*, 2013, pp. 2421–2428.
- [118] L. Janson, E. Schmerling, A. Clark, and M. Pavone, “Fast marching tree: A fast marching sampling-based method for optimal motion planning in many dimensions,” 2013.
- [119] J. D. Gammell, S. S. Srinivasa, and T. D. Barfoot, “Batch informed trees (BIT): Sampling-based optimal planning via the heuristically guided search of implicit random geometric graphs,” in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, May 2015.
- [120] S. Choudhury, J. D. Gammell, T. D. Barfoot, S. S. Srinivasa, and S. A. Scherer, “Regionally accelerated batch informed trees (rabit*): A framework to integrate local information into optimal path planning,” *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 4207–4214, 2016.

- [121] N. Ratliff, M. Zucker, J. A. Bagnell, and S. Srinivasa, “Chomp: Gradient optimization techniques for efficient motion planning,” in *2009 IEEE International Conference on Robotics and Automation*, 2009, pp. 489–494.
- [122] R. Bonalli, A. Cauligi, A. Bylard, and M. Pavone, “Gusto: Guaranteed sequential trajectory optimization via sequential convex programming,” 2019.
- [123] T. Lew, R. Bonalli, and M. Pavone, “Chance-constrained sequential convex programming for robust trajectory optimization,” in *2020 European Control Conference (ECC)*, 2020, pp. 1871–1878.
- [124] S. Primatesta, A. Pagliano, G. Guglieri, and A. Rizzo, “Model predictive sample-based motion planning for unmanned aircraft systems,” in *2021 International Conference on Unmanned Aircraft Systems (ICUAS)*, 2021, pp. 111–119.
- [125] B. Ichter and M. Pavone, “Robot motion planning in learned latent spaces,” *IEEE Robotics and Automation Letters*, vol. 4, no. 3, pp. 2407–2414, 2019.
- [126] K. Lee, D. Choi, and D. Kim, “Incorporation of potential fields and motion primitives for the collision avoidance of unmanned aircraft,” *Applied Sciences*, vol. 11, no. 7, 2021.
- [127] B. Etkin, *Dynamics of Atmospheric Flight*. Courier Corporation, 2012.
- [128] Z. Littlefield, “Efficient and asymptotically optimal kinodynamic motion planning,” Ph.D. dissertation, Rutgers University, 2020.
- [129] B. Nurimbetov, O. Adiyatov, S. Yeleu, and H. A. Varol, “Motion planning for hybrid UAVs in dense urban environments,” in *2017 IEEE International Conference on Advanced Intelligent Mechatronics (AIM)*, 2017, pp. 1627–1632.
- [130] S. M. Beedie, C. M. Harris, J. A. Verberne, C. Y. Justin, and D. Mavris, “Modeling framework for identification and analysis of key metrics for trajectory energy management of electric aircraft,” in *AIAA AVIATION 2021 FORUM*.
- [131] J. A. Verberne, S. M. Beedie, C. M. Harris, C. Y. Justin, and D. N. Mavris, “Development of a simulation environment to identify key metrics to support cockpit decision-making and trajectory energy management systems of electric ctol and vtol vehicles,” in *AIAA AVIATION 2022 Forum*.
- [132] *Collaborative Data Science*. Montreal, QC: Plotly Technologies Inc., 2015.
- [133] J. D. Hunter, “Matplotlib: A 2d graphics environment,” *Computing in Science & Engineering*, vol. 9, no. 3, pp. 90–95, 2007.

- [134] E. Frazzoli, “Robust hybrid control for autonomous vehicle motion planning,” Ph.D. dissertation, Massachusetts Institute of Technology, 2001.
- [135] M. W. Mueller, M. Hehn, and R. D’Andrea, “A computationally efficient motion primitive for quadrocopter trajectory generation,” *IEEE Transactions on Robotics*, vol. 31, no. 6, pp. 1294–1310, 2015.
- [136] T. Zhiling, B. Chen, R. Lan, and S. Li, “Vector field guided RRT* based on motion primitives for quadrotor kinodynamic planning,” *Journal of Intelligent & Robotic Systems*, vol. 100, Dec. 2020.
- [137] T. Schouwenaars, B. Mettler, E. Feron, and J. How, “Robust motion planning using a maneuver automation with built-in uncertainties,” vol. 3, Jul. 2003, 2211–2216 vol.3, ISBN: 0-7803-7896-2.
- [138] M. Kamel, M. Burri, and R. Y. Siegwart, “Linear vs nonlinear MPC for trajectory tracking applied to rotary wing micro aerial vehicles,” *ArXiv*, vol. abs/1611.09240, 2016.
- [139] T. A. Rodrigues, J. Patrikar, N. L. Oliveira, H. S. Matthews, S. Scherer, and C. Samaras, *Drone flight data reveal energy and greenhouse gas emissions savings for small package delivery*, 2021.
- [140] T. A. Rodrigues, J. Patrikar, B. Oliveira Wagner, S. Scherer, and C. Samaras, *Development of an energy model for quadcopter package delivery drones*, 2021.
- [141] J. Stolaroff, C. Samaras, E. O’Neill, A. Lubers, A. Mitchell, and D. Ceperley, “Energy use and life cycle greenhouse gas emissions of drones for commercial package delivery,” *Nature Communications*, vol. 9, Feb. 2018.
- [142] G. Hoffmann, H. Huang, S. Waslander, and C. Tomlin, “Quadrotor helicopter flight dynamics and control: Theory and experiment,” in *AIAA Guidance, Navigation and Control Conference and Exhibit*.
- [143] A. Sharma, “System identification of a micro aerial vehicle,” Master’s Project, Luleå University of Technology, 2019.
- [144] T. A. Rodrigues *et al.*, “In-flight positional and energy use data set of a DJI matrice 100 quadcopter for small package delivery,” *Scientific data*, vol. 8, no. 1, p. 155, Jun. 2021.
- [145] M. Pivtoraiko and A. Kelly, “Kinodynamic motion planning with state lattice motion primitives,” in *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2011, pp. 2172–2179.

- [146] O. Ljungqvist, N. Evestedt, M. Cirillo, D. Axehill, and O. Holmer, “Lattice-based motion planning for a general 2-trailer system,” Jun. 2017.
- [147] V. K. Viswanathan, E. Dexheimer, G. Li, G. Loianno, M. Kaess, and S. Scherer, “Efficient trajectory library filtering for quadrotor flight in unknown environments,” in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020, pp. 2510–2517.
- [148] Y. Tassa, N. Mansard, and E. Todorov, “Control-limited differential dynamic programming,” May 2014.
- [149] H. Lee, C. M. Harris, J. C. Gladin, and D. N. Mavris, “A method for simultaneous optimization of power split and flight path trajectories for hybrid electric aircraft,” in *AIAA Scitech 2021 Forum*.
- [150] G. Best, R. Garg, J. Keller, G. Hollinger, and S. Scherer, “Resilient multi-sensor exploration of multifarious environments with a team of aerial robots,” May 2022.
- [151] X. Xiao, J. Dufek, and R. R. Murphy, “Robot risk-awareness by formal risk reasoning and planning,” *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 2856–2863, 2020.
- [152] X. Xiao, “Risk-aware path and motion planning for a tethered aerial visual assistant in unstructured or confined environments,” Ph.D. dissertation, Texas A&M University, 2019.
- [153] J. A. Montoya-Zegarra, J. D. Wegner, L. Ladický, and K. Schindler, “Semantic segmentation of aerial images in urban areas with class-specific higher-order cliques,” in *ISPRS Ann. Photogramm. Remote Sens. Spatial Inf. Sci.*, vol. II-3/W4, Copernicus Publications, pp. 127–133.
- [154] S. Singh, “Perception for safe autonomous helicopter flight and landing,” in *American Helicopter Society, Forum 72*, May 2016.
- [155] J. Kim, C. Zhang, S. Briceno, and D. Mavris, “Supervised machine learning-based wind prediction to enable real-time flight path planning,” Jan. 2021.
- [156] F. S. Barbosa, B. Lacerda, P. Duckworth, J. Tumova, and N. Hawes, *Risk-aware motion planning in partially known environments*, 2021.
- [157] J. Dong, M. Mukadam, F. Dellaert, and B. Boots, “Motion planning as probabilistic inference using gaussian processes and factor graphs,” Jun. 2016.

- [158] V. D. Sharma, M. Toubeh, L. Zhou, and P. Tokekar, “Risk-aware planning and assignment for ground vehicles using uncertain perception from aerial vehicles,” 2020.
- [159] R. T. Rockafellar and S. Uryasev, “Optimization of conditional value-at-risk,” *Journal of Risk*, vol. 2, pp. 21–41, 2000.
- [160] M. Norton, V. Khokhlov, and S. Uryasev, “Calculating cvar and bpoe for common probability distributions with application to portfolio optimization and density estimation,” *Annals of Operations Research, Special Issue: Recent Developments in Financial Modeling and Risk Management*, vol. 299, pp. 1281–1315, Oct. 2018.
- [161] H. Zhang, Y. Yao, K. Xie, C.-W. Fu, H. Zhang, and H. Huang, “Continuous aerial path planning for 3d urban scene reconstruction,” *ACM Trans. Graph.*, vol. 40, no. 6, Dec. 2021.
- [162] R. Pěnička, J. Faigl, and M. Saska, “Physical orienteering problem for unmanned aerial vehicle data collection planning in environments with obstacles,” *IEEE Robotics and Automation Letters*, vol. 4, no. 3, pp. 3005–3012, 2019.
- [163] R. Pěnička, J. Faigl, M. Saska, and P. Váňa, “Data collection planning with non-zero sensing distance for a budget and curvature constrained unmanned aerial vehicle,” *Autonomous Robots*, vol. 43, Dec. 2019.
- [164] F. Kurz, D. Rosenbaum, J. Leitloff, O. Meynberg, and P. Reinartz, “Real time camera system for disaster and traffic monitoring,” May 2011.
- [165] A. Jasour, “Risk aware planning and control of probabilistic nonlinear dynamical systems,” 2019.
- [166] V. Sharma, M. Toubeh, L. Zhou, and P. Tokekar, “Risk-aware planning and assignment for ground vehicles using uncertain perception from aerial vehicles,” Mar. 2020.
- [167] X. Dong, H. Lu, Y. Xia, and Z. Xiong, “Decision-making model under risk assessment based on entropy,” *Entropy*, vol. 18, no. 11, 2016.
- [168] A. Gelman and Y. Yao, “Holes in bayesian statistics,” *Journal of Physics G: Nuclear and Particle Physics*, vol. 48, no. 1, p. 014 002, Dec. 2020.
- [169] A. Gelman, D. Simpson, and M. Betancourt, “The prior can often only be understood in the context of the likelihood,” *Entropy*, vol. 19, no. 10, 2017.

- [170] L. K. Newman, R. C. Frigm, M. G. Duncan, and M. D. Hejduk, “Evolution and implementation of the nasa robotic conjunction assessment risk analysis concept of operations,” 2014.
- [171] O. Arslan and P. Tsiotras, “Machine learning guided exploration for sampling-based motion planning algorithms,” in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2015, pp. 2646–2652.
- [172] Z. Li, J. Wang, and M. Q. Meng, “Efficient heuristic generation for robot path planning with recurrent generative model,” *CoRR*, vol. abs/2012.03449, 2020. arXiv: 2012.03449.
- [173] J. Wang, W. Chi, C. Li, C. Wang, and M. Q. Meng, “Neural RRT*: Learning-based optimal path planning,” *IEEE Transactions on Automation Science and Engineering*, vol. 17, no. 4, pp. 1748–1758, 2020.
- [174] J. D. Gammell, T. D. Barfoot, and S. S. Srinivasa, “Informed sampling for asymptotically optimal path planning,” *IEEE Transactions on Robotics*, vol. 34, no. 4, pp. 966–984, Aug. 2018.
- [175] J. D. Gammell, S. S. Srinivasa, and T. D. Barfoot, “Informed RRT: Optimal sampling-based path planning focused via direct sampling of an admissible ellipsoidal heuristic,” in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, Sep. 2014.
- [176] Z. Littlefield and K. E. Bekris, “Informed asymptotically near-optimal planning for field robots with dynamics,” in *FSR*, 2017.
- [177] —, “Efficient and asymptotically optimal kinodynamic motion planning via dominance-informed regions,” in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018, pp. 1–9.
- [178] A. Sivaramakrishnan, Z. Littlefield, and K. E. Bekris, “Towards learning efficient maneuver sets for kinodynamic motion planning,” 2019.
- [179] B. Moon, S. Chatterjee, and S. Scherer, “Tigris: An informed sampling-based algorithm for informative path planning,” 2022.
- [180] D. T. Larsson, D. Maity, and P. Tsiotras, “A linear programming approach for resource-aware information-theoretic tree abstractions,” 2022.
- [181] J. Chen, “Washington dc 3d city map for 3D urban UAV relay placement,” 2020.
- [182] M. Zoula, M. Prágr, and J. Faigl, “On building communication maps in subterranean environments,” in Mar. 2021, pp. 15–28, ISBN: 978-3-030-70739-2.

- [183] C. M. Harris, M.-D. R. Sokollek, L. S. Nunez, J. T. Valco, M. Balchanos, and D. N. Mavris, “Simulation-based uas swarm selection for monitoring and detection of migrant border crossings,” in *2018 Aviation Technology, Integration, and Operations Conference*.
- [184] M. Chandarana, D. Hughes, M. Lewis, K. Sycara, and S. Scherer, “Planning and monitoring multi-job type swarm search and service missions,” *Journal of Intelligent & Robotic Systems*, vol. 101, no. 3, p. 44, 2021.
- [185] W. Brogan, *Modern Control Theory*, ser. QPI series. Quantum Publishers, 1974, ISBN: 9780135903070.
- [186] D. P. Bertsekas, *Dynamic Programming and Optimal Control*, 4th. Belmont, MA, USA: Athena Scientific, 2017, vol. I.
- [187] A. Girard, C. Guernic, and O. Maler, “Efficient computation of reachable sets of linear time-invariant systems with inputs,” vol. 3927, Mar. 2006, pp. 257–271.

VITA

Caleb Harris was born and raised near Memphis, TN and attended the University of Memphis where he received a Bachelor of Science in Mechanical Engineering in 2017. He then moved to Atlanta Georgia to attend Georgia Institute of Technology and worked under Dr. Dimitri Mavris in the Aerospace Systems Design Laboratory. He received a MS in Aerospace Engineering degree in December 2019, a MS in Computational Science and Engineering in May 2021, and will receive a PhD in Computational Science and Engineering in 2022. His work is focused on data-driven analysis and efficient trajectory planning for improving the safety of aerial systems when operating in complex environments. He is interested in how these methods can improve aerial system search and rescue, disaster response, and urban air mobility.