Metalevel Motion Planning for Unmanned Aircraft Systems: Metrics Definition and Algorithm Selection

by

Cosme A. Ochoa

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Robotics)
in The University of Michigan
2021

Doctoral Committee:

Professor Ella M. Atkins, Chair
Professor Jessy W. Grizzle
Professor Odest C. Jenkins
Professor Ilya V. Kolmanovsky

First, it is fun to try to get machines to solve problems for which even humans have great difficulty. Second, planning algorithms have achieved widespread successes in several industries and academic disciplines, including robotics, manufacturing, drug design, and aerospace applications. The rapid growth in recent years indicates that many more fascinating applications may be on the horizon.

— Dr. Steven M. LaValle

Cosme A. Ochoa
cosme@umich.edu
ORCID iD: 0000-0002-8622-8326

© Cosme A. Ochoa 2021

To my parents, for all your sacrifices.

To my friends, for when things got rough.

To my future self, what a journey.

ACKNOWLEDGEMENTS

The past five years have been a rollercoaster of a journey. It had its ups and downs, but we have finally reached its satisfying conclusion through hard efforts and invaluable friendships. At the time of this writing, we are facing the most challenging crisis in recent history worldwide. Despite all political, social, health, economic, and inner turmoils, the day will come when things turn for the better, and we put all this behind us.

During my time at Michigan, many friendships were made throughout the years that one hopes will last a lifetime. A first-generation minority Ph.D. student more than 1600 miles away from home, I cannot describe the amazing help from fellow colleagues Pedro di Donato and Swee Balachandran during my early years. They welcomed an outsider with open arms and shared countless nights with me as I prepared for candidacy and themselves for graduation. During my mid-tenure, Brian Yao, Chen Li, Jeremy Castagno, and Prashin Sharma were and are a joy to work with. Anytime they showed up to our office or lab made my day. Lastly, during the end of my academic career, Prince Kuevor, Matthew Romano, Akshay Mathur, and Joseph Kim kept the kid in me alive with countless fun gatherings into the break of dawn. Pha, Chenlan, and Daniel thank you for making life outside work a treat. Y'all are not just my friends; y'all are my second family.

Two distinguished individuals without whom this academic endeavor would not have been possible and deserve special recognition are Katja Meuche and Ella Atkins. Katja, while we started as mentor and mentee mentee, you rather quickly became one

of my best friends. Our time working together was noteworthy due to its amazing productivity and indispensable trust, venturing outside our research comfort zones. Ella was more than an advisor. She was a friend and, at times, a mother-figure when needed. Juggling her multiple hats, she always made herself available and could be trusted to be just and frank. Especially during my final year, Ella went far beyond her role to help me get through some very rough times. Thank you, Ella, we made it, and I am honored to have been your student.

Lastly, to my family. During the past ten years, I have been away for its majority. Holidays were skipped, birthdays were missed, and relatives now watch me from the heavens, but we can now see it was worth it. It is crazy to think how different my life would have been if I had been born on another side of a river. Many sacrifices were made along the way but Esmeralda and José you raised a decent gentleman and now doctor despite our rough circumstances. While you were without your big brother for a decent portion of your childhoods, Emily and Christian, I hope I made you proud.

TABLE OF CONTENTS

LIST OF FIGURES

Figure

LIST OF TABLES

Table

LIST OF APPENDICES

Appendix

LIST OF ABBREVIATIONS

2D Two Dimensional

3D Three Dimensional

AABB Axis Aligned Bounding Box

ACC Accuracy

AGL Above Ground Level

AIT Adaptively Informed Trees

AR Anytime Repairing

ASP Algorithm Selection Problem

ATC Air Traffic Control

BEST Best Linear Unbiased Estimator

BIT Batched Informed Trees

BVLOS Beyond Visual Line-of-Sight

CRS Coordinate Reference System

DOD Department of Defense

DOT Department of Transportation

FAA Federal Aviation Administration

FMT Fast Marching Trees

GDAL Geospatial Data Abstraction Library

GDOP Geometric Dilution of Precision

GIS Geographic Information System

GNSS Global Navigation Satellite System

GNC Guidance, Navigation, and Control

GPS Global Positioning System

IQR Interquartile Range

LIDAR Light Detection and Ranging

LP Lifelong Planning

MAE Mean Absolute Error

NYC New York City

OSM OpenStreetMap

PDOP Position Dilution of Precision

PRM Probabilistic Roadmap

PTP Point to Point

RC Radio Control

RRT Rapidly-exploring Random Trees

SAT Satisfiability

SLAM Simultaneous Localization and Mapping

SSP Sensor Selection Problem

STC Spanning Tree Coverage

STD Standard Deviation

UAS Unmanned Aircraft Systems

UAM Urban Air Mobility

UERE User Equivalent Range Error

UTM Universal Transverse Mercator

TDOP Time Dilution of Precision

LIST OF SYMBOLS

x, y, z	aircraft position (intertial frame)
$\mathcal{A}$	algorithm portfolio
$\mathcal{P}$	problem instance set
$\hat{\mathcal{P}}$	problem instance subset
a	algorithm
p	problem instance
$\mathcal{D}$	date-time information
$\mathcal{I}$	aircraft information
$\mathcal{L}$	total bounding box
$\mathcal{W}$	weighting vector
w_{dist}	distance metric weight
w_{gps}	GPS metric weight
w_{lidar}	Lidar metric weight
w_{pop}	population metric weight
w_{risk}	risk metric weight
$\mathcal{H}$	metric/cost map
$\mathcal{H}_{total}$	total cost map
$\mathcal{H}_{norm}$	min-max normalized map
δ_{view}	sensor field of view/range
δ_{res}	map resolution
δ_b	bounding box buffer distance
m_p	path-based metrics
m_m	map-based metrics

m_s	software-based metrics
m_{gps}, c_{gps}	GPS pseudorange uncertainty metric/cost
m_{lidar}, c_{lidar}	Lidar-based visibility metric/cost
m_{obs}, c_{obs}	obstacle occupancy metric/cost
m_{pop}, c_{pop}	overflown population density metric/cost
m_{risk}, c_{risk}	obstacle proximity metric/cost
m_{dist}, c_{dist}	distance traveled metric/cost
m_{time}, c_{time}	algorithm execution time metric/cost
m_{mem}, c_{mem}	algorithm memory usage metric/cost
c	speed of light
ρ_{rc}	GPS receiver range
t_{rc}	GPS receiver clock
t_{sat}	GPS satellite clock
ϵ_{UERE}	UERE catch-all error term
x_{rc}, y_{rc}, z_{rc}	GPS receiver position (inertial frame)
x_{sat}, y_{sat}, z_{sat}	GPS satellite position (inertial frame)
N_{sats}	number of visible satellites
G_{gps}	GPS pseudorange linear system
$\boldsymbol{\chi}_{gps}$	GPS pseudorange state vector
Σ_{gps}	GPS pseudorange system covariance
$GDOP_{thresh}$	GDOP threshold
$b_{\mathcal{O}}$	Lidar beams' origin
b_{lidar}	number of Lidar beams
f_{lidar}	number of Lidar scanning positions
n_{lidar}	maximum Lidar returns per revolution
Ω_{obs}	obstacle set
r_{lidar}	Lidar visible range
s_{lidar}	number of Lidar scan returns
β	Lidar beam elevation angle
$\mathcal{C}_{free}$	obstacle-free configuration space

$\mathcal{C}_{obs}$	obstacle configuration space
$\mathcal{C}_{tot}$	total configuration space
pop_{census}	census population count
$\hat{pop}_{norm}$	maximum population count
Γ	daytime population modifier
Γ_{comm}	commercial area population modifier
Γ_{resi}	commercial area population modifier
$\mathcal{B}$	operating bounding box
$\hat{\mathcal{B}}$	buffered operating bounding box
d_{close}	distance to closest obstacle surface
d_{thresh}	proximity risk distance threshold
t_0, t_f	flight initial and final time
v	vehicle velocity
$\boldsymbol{\zeta}$	flight plan/path
$t_{a,0}, t_{a,f}$	algorithm initial and final execution times
Π	memory usage supervisor
$\mathcal{H}_{obs}$	obsacle occurpancy map
$\mathcal{H}_{gps}$	GPS uncertainty map
$\mathcal{H}_{lidar}$	Lidar visibility map
$\mathcal{H}_{pop}$	population density map
$\mathcal{H}_{risk}$	proximity risk map
δ_z	mapping alititude
δ_r	mapping resolution
$\mathcal{S}$	ASP selection function
$\mathcal{A}^*$	ranked algorithm porfolio
a^*	selected algorithm
$\mathcal{Q}_S$	start vehicle state
$\mathcal{Q}_G$	goal vehicle state
idx, idy, idz	state indices
d_{euc}	Eucledian distance

d_{oct}	octile distance
f	total path cost estimate
g	*cost-so-far* function
h	*cost-to-go* heursitic function
h_{dist}	Eucledian distance heuristic
h_{plus}	weighted multi-objective heuristic
$\mathcal{G}$	search graph
T	search tree
V	search nodes
E	search edges
sr	success rate
$Q1, Q3$	first and third data quartiles
r_{min}, r_{max}	aircraft range bounds
$\mathcal{T}_{cost}$	normalized mean planning set cost
$\mathcal{T}_{success}$	normalized mean planning set success rate
T_C, T_S	cost and success-based ASP decision trees
D	tree decision node
X_{train}, Y_{train}	training inputs and labels
$\mathcal{Z}$	latent map set
η	latent map respresentation
$\mathcal{E}$	latent encoder function
$\boldsymbol{x}_{in}$	formatted network input
$\boldsymbol{y}_{in}$	formatted network output
N_C	cost estimator neural network
N_S	success predictor neural network
N_H, N_U	hybrid and unified ASP neural networks
$\mathcal{S}_H, \mathcal{S}_U$	hybrid and unified ASP selections
$\hat{c}$	network cost score estimate
$\hat{\rho}$	network success likehood prediction

ABSTRACT

A diverse suite of manned and unmanned aircraft will occupy future urban airspace. Flight plans must accommodate specific aircraft characteristics, including physical volume with safety zone clearance, landing/takeoff procedures, kinodynamics, and a wide range of flight environments. No single motion planner is applicable across all possible aircraft configurations and operating conditions. This dissertation proposes the first motion planning algorithm selection capability with application to small Unmanned Aircraft System (UAS) multicopters operating in and over a complex urban landscape.

Alternative data-driven fail-safe protocols are presented to improve on contemporary "fly-home" or automatic landing protocols, focusing on rooftops as safe urban landing sites. In a fail-safe direct strategy, the multicopter identifies, generates, and follows a flight plan to the closest available rooftop suitable for landing. In a fail-safe supervisory strategy, the multicopter examines rooftops en route to a planned landing site, diverting to a closer, clear landing site when possible. In a fail-safe coverage strategy, the multicopter cannot preplan a safe landing site due to missing data. The multicopter executes a coverage path to explore the area and evaluate overflown rooftops to find a safe landing site. These three fail-safe algorithms integrate map generation, flight planning, and area coverage capabilities.

The motion planning algorithm selection problem (ASP) requires qualitative and quantitative metrics to inform the ASP of user/agent, algorithm, and configuration space preferences and constraints. Urban flight map-based, path-based, and software-based cost metrics are defined to provide insights into the urban canyon properties

needed to construct safe and efficient flight plans. Map-based metrics describe the operating environment by constructing a collection of GPS/Lidar navigation performance, population density, and obstacle risk exposure metric maps. Path-based metrics account for a vehicle's energy consumption and distance traveled. Software-based metrics measure memory consumption and execution time of an algorithm. The proposed metrics provide pre-flight insights typically ignored by obstacle-only planning environment definitions.

An algorithm portfolio consisting of geometric point-to-point (PTP), graph-based (A^*_{dist}, A^*_{plus}), and sampling-based (BIT^*_{dist}, BIT^*_{plus}) motion planners was considered. Path cost, execution time, and success rate benchmarks were investigated using Monte Carlo problem instances with A^*_{plus} producing the lowest cost paths, PTP having the fastest execution times, and A^*_{dist} having the best overall success rates. BIT^* paths typically had higher cost, but their success rate increased relative to altitude. Problem instances and metric maps informed two new machine learning solutions for urban small UAS motion planning ASP. ASP decision trees were simple to construct but unable to capture both complex cost metrics and algorithm execution properties. Neural network-based ASP formulations produced promising results, with a hybrid two-stage selection scheme having the best algorithm selection accuracy. Solving the motion planning ASP allows for data-driven motion planner execution cognizant of available resources and environment properties that may change in space and time.

The most significant innovation of this dissertation is motion planning ASP for UAS. Non-traditional open-source databases also advance the field of data-driven flight planning, contributing to fail-safe UAS operations as well as ASP. Path planning algorithms integrated a new suite of diverse cost metrics accompanied by a novel multi-objective admissible heuristic function. Neural network and decision tree ASP options were presented and evaluated as a first-case practical approach to solving the motion planning ASP for small UAS urban flight.

CHAPTER I

Introduction

The proliferation of small unmanned aircraft systems (UAS) has fueled the need for safe, efficient, and cost-effective flight hardware-software systems certified for autonomous operations. The US Department of Transportation (DOT) forecasts the deployment of 14000 UAS by the Department of Defense (DOD), 70000 UAS by local and state governments, and 175000 UAS in the commercial marketplace by 2035 [6]. As early as 2016, autonomous beyond visual line-of-sight (BVLOS) small UAS test flights in rural and suburban areas had been conducted by commercial entities [7]. As these technologies mature and become mainstream, rigorous hardware, software, and systems certification standards are required to enable routine BVLOS autonomous UAS deployment, especially over populated urban centers, and autonomous passenger-carrying Urban Air Mobility (UAM) operations [8].

UAS platforms expected to populate the urban airspace include (un)manned fixed-wing, rotorcraft, multicopter (drone), air taxis, and other hybrid aircraft. Different aerial vehicle configurations impose different landing site constraints [9, 10, 11]. Flight plans must accommodate specific UAS characteristics, including physical volume with safety zone clearance, landing/takeoff procedures, kinodynamics, and a wide range of flight environments. All autonomous UAS require a trusted motion planning capability, yet no single tactical/motion planner is applicable across all possible UAS

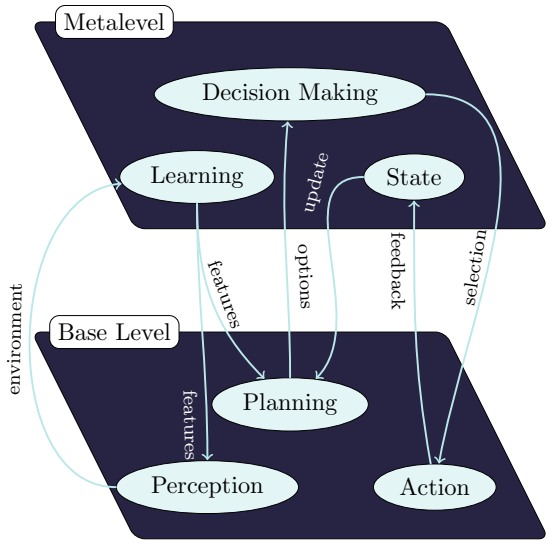


Figure 1.1: Overview of metalevel framework for a general robotic system.

configurations and operating conditions. Decoupling [12, 13, 14] and multi-stage approaches [15, 16, 17] to motion planning have been employed when no single algorithm will suffice. Such approaches provide insight into the practical need for multiple algorithms to solve different motion planning problems. However, the available motion planning literature does not yet offer a general algorithm selection methodology. This dissertation proposes the first motion planning algorithm selection capability with application to small UAS operating in and over a complex urban landscape.

A baseline autonomous system must be capable of sensing its environment, planning a course of action, and executing that plan. Under dynamic, uncertain conditions, learning and decision making typically performed by a human supervisor require autonomy with metalevel reasoning as shown in Fig. 1.1. This dissertation is concerned with metalevel "decision making" for motion planning algorithm selection. In particular, algorithm selection is a metalevel decision scheme governing motion planning "options" executed at the base level.

In computer science, SATzilla [18] was an early algorithm selection application to satisfiability (SAT) using empirical hardness models for automated algorithm portfolio construction. The success of SATzilla in the 2007 SAT Competition [19] gave

rise to follow-on algorithm selection work utilizing latent class models [20], k-nearest neighbors [21], and hierarchical clustering [22]. Complementary work in algorithm configuration [23] and scheduling [24] selection has also been motivated by success in the SAT community.

Beyond SAT, the value of algorithm selection has also been recognized for automated task planning applications. Learning strategies have exploited learned algorithm performance models [25] and planning domain macro encodings [26, 27] to inform the portfolio configuration and scheduling process. Online approaches [28, 29] have used regression-based strategies to use new data instances to improve their selection strategy periodically. This thesis addresses the urban flight planning problem [30] for small UAS with a focus on motion planning algorithm selection [31].

1.1 Problem Statement

The diversity of small UAS missions, aircraft, and overflight landscapes motivates the motion planning problem studied in this dissertation: *Given all available information, e.g., environment maps, mission constraints, and aircraft specifications, what is the most suitable flight planning strategy?*

This algorithm selection problem (ASP) [32] describes the notion of selecting the best performing or most suitable algorithm for a particular problem instance. Rule-based [33] and data-driven [34] ASP techniques have been developed. Rule-based approaches offer a simple structured framework but are limited by rigidity and can exhibit inherent bias [35, 36]. Data-driven or meta-learning approaches [37] can identify features of interest to construct decision-making models with regression and unsupervised clustering techniques. To our knowledge, this dissertation is the first work to develop and evaluate an algorithm selection capability for motion planning using a data-driven (meta-learning) approach.

A motion planner sequences viable motion primitives to reach a target destination

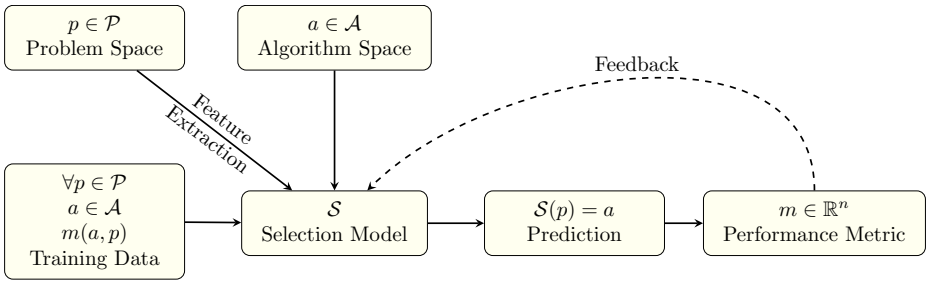


Figure 1.2: Algorithm selection problem (ASP) as defined in [1].

or accomplish a desired task, e.g., search & rescue [38, 39], reconnaissance [40, 41], sense & avoid [42, 43], navigating uncertain environments [44, 45, 46]. Motion planners exploit real-time sensing to manage risk [11], resolve conflicts [47], and offer fail-safe solutions [48].

Motion planners offer different benefits [49, 50, 51, 52, 53]. Geometric planners are usually fast but may not avoid obstacles or minimize desired costs. Graph-based planners minimize cost but are computationally complex, an issue addressed in part with sampling-based planners. Sampling-based planners efficiently navigate sparse obstacle environments, but graph-based planners tend to outperform in cluttered spaces. Optimal control solvers assure physical constraints are satisfied but are complex and experience local minima issues [31]. This dissertation defines and addresses the motion planning ASP for small UAS UAM.

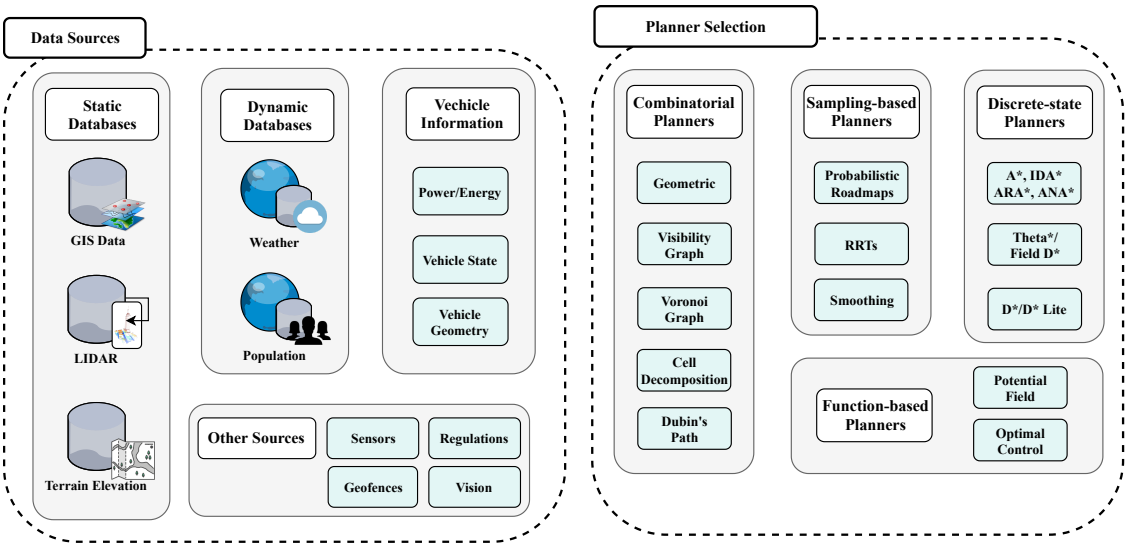
Let $\mathcal{A}$ be a portfolio of motion planning algorithms available to solve a set of flight planning problem instances $\mathcal{P}$ for an autonomous small UAS (sUAS). For each problem instance $p \in \mathcal{P}$, there exists an algorithm $a \in \mathcal{A}$ most suitable for solving the given problem instance. This dissertation proposes an algorithm selection ranking function $\mathcal{S} : \mathcal{P} \mapsto \mathcal{A}^*$ to create a planner preference ranking $\mathcal{A}^*$ for each planning instance p without the need to run all algorithms in $\mathcal{A}$ at run-time. $\mathcal{S}$ orders all planners expected to meet specified time and memory constraints in $\mathcal{A}^*$ to determine such a^*, the motion planner expected to perform best for problem instance p as depicted in Fig. 1.2 [1].

A problem instance $p \in \mathcal{P}$ is defined over the septuple $< \mathcal{B}, \mathcal{Q}_S, \mathcal{Q}_G, \mathcal{D}, \mathcal{H}, \mathcal{I}, \mathcal{W} >$, where $\mathcal{B}$ is a designated operating bounding box, $\mathcal{Q}_S$ & $\mathcal{Q}_G$ are the start and goal configurations, $\mathcal{D}$ are date and time details, $\mathcal{H}$ is an environment model or map, $\mathcal{I}$ is the aircraft information (type and properties), and $\mathcal{W}$ is a set of cost metric weightings. $\mathcal{S}$ relies on three classes of cost and constraint metrics: path-based m_p, map-based m_m, and software execution-based m_s, to inform the selection of an algorithm.

1.2 Approach

To develop a capable motion planning ASP solution for sUAS, three related problems are explored: (1) Fail-safe UAS emergency landing planning scenarios to motivate and understand key costs and constraints (Chapter 2), (2) Map-based metric definitions to inform UAS ASP and motion planners (Chapter 3), and (3) Decision tree and neural network-based ASP solvers trained and evaluated on real-world map data (Chapter 4). To explore fail-safe planning, a Manhattan (New York) map is translated to obstacle and cost maps used by a UAS to define safe landing plans. A suite of vehicle (path-based) and environment (map-based) cost metrics are defined to guide motion planning. Vehicle energy consumption (fuel or battery), distance traveled, and risk exposure are to be minimized. Map-based metrics describe navigation, population, building, and terrain costs. Software-based metrics (memory and computation time) evaluate real-time (in situ) planning potential. Given the diverse nature of vehicular platforms and environment-specific characteristics, a plethora of motion planning algorithms and data sources can be used as illustrated in Figs. 1.3a and 1.3b.

The algorithm portfolio $\mathcal{A}$ selected for this work covers the following motion planning categories: geometric, discrete/graph-based, and iterative/sampling-based. Geometric flight planning is a staple of aerial motion planning. Dubins curves [54] for fixed-wing aircraft and point-to-point (PTP) flight plans for multicopters are com-



(a) Potential data sources in an urban environment for data-driven motion planning.

(b) Categorization of viable motion planners to select from for guidance and navigation.

Figure 1.3: Categorization of metalevel planning data sources and motion planners.

mon geometric planners used in obstacle-free environments. Of the many discrete search planners, A* [55] is recognized for its efficient search combining cost with an admissible heuristic to generate an optimal path. Similarly, BIT* [3], inspired by rapidly-exploring random tree algorithms RRT* [56] and Informed RRT [57], efficiently returns an optimal solution by iteratively reducing the search space. To the author's best knowledge and at the time of this work, the presented motion planning algorithms are competitive in their respective categories.

We define two decision trees and two neural network architectures for urban ASP. Each decision tree uses a total cost-based or success-based likelihood metric to select the best planning algorithm. Each neural network uses encoded map feature vectors and a cost weight vector $\mathcal{W}$ to predict expected performance for each planner given each overflight region. In the first scenario, a neural network is defined for each planner in the algorithm portfolio to estimate that planner's expected performance. Planner options are then ranked based on expected plan cost and computation overhead estimates. The second neural network architecture outputs a single "best"

algorithm choice based on path, map, weight, and problem instance network inputs.

Planning instances are randomly generated over a 20 km x 10 km Manhattan (New York) region $\mathcal{L}$ for a multicopter small UAS. Start/goal configurations and cost weight vectors are randomly sampled. For each algorithm $a \in \mathcal{A}$, a weighted path cost is calculated, if path construction is feasible, subject to $\mathcal{W}_k, \boldsymbol{\mathcal{H}}_k$, and software execution metrics $m_{s,k}$ are recorded for training purposes. Decision tree and neural network results are evaluated against truth data and each other for algorithm selection accuracy.

1.3 Contributions and Innovations

Contributions of this dissertation are:

- Fail-safe UAS urgent landing planning options for urban flight.
- Non-traditional open-source database use for data-driven flight planning.
- Path planning algorithm adaptation to diverse cost metrics.
- Monte Carlo simulations to generate training and test data.
- Benchmark evaluation of neural network and decision tree ASP options.

Innovations of this dissertation are:

- A fail-safe planner that exploits map and real-time sensor inputs to guide small UAS urgent landing in a complex urban setting.
- Definition of novel metalevel planning metrics to guide ASP as well as flight planning.
- Definition of a novel multi-objective heuristic function for search-based and sampling-based path planners.
- Definition of diverse decision trees as simple rule-based options for motion planning selection.
- Adaptation of neural network designs to rapidly solve the motion planning ASP.

1.4 Outline

The remainder of the document is structured as follows. Chap. II introduces the use of non-traditional databases for data-driven flight planning. With a focus on urban emergency landings, this chapter investigates three alternatives informed failsafe protocols against the "return-to-home" protocol available by default on most consumer unmanned aviation platforms. Chap. III follows this work and expands the use of databases to the general global urban flight planning problem. A set of map-based, path-based, and software-based metrics is defined to inform a flight planner's selection beyond traditional motion planning metrics. Chap. IV analyzes the motion planning ASP in detail and examines rule-based and data-driven selection approaches. Using the metrics defined in Chap. III, a pair of neural network-based selection schemes are presented to select the most suitable flight planner trading off map-based and path-based metrics while abiding by software-based constraints. Chap. V presents conclusions from this work and proposes future work avenues left to mature and transition data-driven urban flight planning with algorithm selection to practical implementation.

CHAPTER II

Data-driven Failsafe Protocols

2.1 Introduction

Small Unmanned Aircraft Systems (sUAS) are expected to fly over populated urban centers and isolated rural test ranges once they are sufficiently safe. Triple redundancy is likely not feasible, so sensor-data fusion and fail-safe contingency response are critical for developing, certifying, and deploying across the emerging sUAS fleet. Subject to real-time and situational awareness decision-making constraints, a remote operator/dispatcher's supervisory reactions may be inadequate for UAS contingency response. Instead, UAS autonomy must be able to assess a hazardous situation and react quickly. Current air traffic control (ATC) procedures do not address UAS failure events, particularly since sUAS are expected to operate with a minimal altitude recovery margin. In addition, UAS rely on datalink-based communication for remote pilot commands to even be received. In the event of a lost data signal, neither ATC nor the remote pilot can command the sUAS, further motivating the use of onboard fail-safe protocols [58].

Current fail-safe protocols in hobby-class multicopter systems respond to preprogrammed behaviors to prevent the crash of an sUAS in response to a recognized anomaly or system failure. Existing fail-safe protocols include returning to a designated home point when the radio control (RC) or datalink signal has been lost or

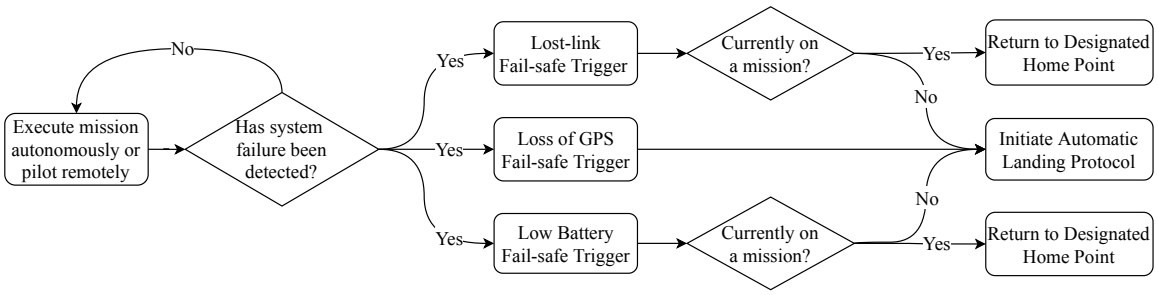


Figure 2.1: Contemporary fail-safe protocols on existing sUAS platforms trigger automatic landing or a return-to-home action.

when the multicopter battery dips below a fixed threshold. If the multicopter loses a navigation (e.g., GPS) signal or detects a critically low battery level, the multicopter will typically initiate an automatic landing protocol [59] as shown in Fig. 2.1. Electronic geofencing has also been proposed to prevent fly-away [60, 61] but to-date has relied on user data entry or a sparse database of restricted sites, e.g., the White House.

The concept of map-based fail-safe sUAS flight planning was first introduced by [2], which used map-based elements to guide a UAS experiencing low energy levels when flying above Manhattan, New York in an emergency landing. Ref. [2] utilized public building, terrain, and population databases to create a cost map explored by an A* search algorithm to generate an emergency flight plan. Initially, database map features can augment the search space typically limited by sensor range and field of view. On final approach, a sensor-based planner can be used to define an obstacle-free landing path as described in 2.2.

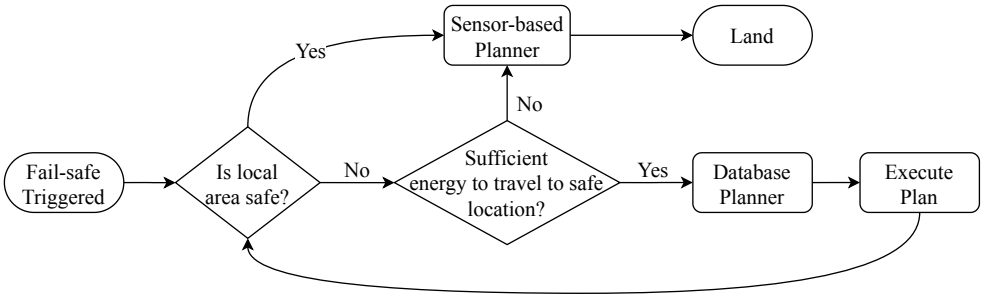


Figure 2.2: Emergency sensor-database planner as adapted from [2].

This chapter investigates three data-driven fail-safe protocols that enable a multicopter UAS experiencing an anomaly to land in a complex urban environment safely. Map features are defined from geospatial databases to make an informed decision regarding where to land, focusing on building rooftops as safe urban landing sites. PLUTO and MapPLUTO land use and geographic databases of Manhattan's tax lots [62] are processed for path planning. While rooftops are not the only emergency landing option, they are perhaps the most unlikely areas to be occupied, thus presenting a relatively low risk of harm to people.

In the first fail-safe strategy, the multicopter identifies nearby buildings designated as potential landing sites, then generates and follows a flight plan to the closest available rooftop suitable for landing. Given no nearby flat open rooftops, the multicopter can fly to a surface landing site, e.g., the nearest open park. In a second "hybrid" fail-safe strategy, the multicopter examines rooftops en route to a planned landing site using onboard imagers and diverts to a closer, clear landing site when possible. In the third fail-safe strategy, the multicopter cannot preplan a safe landing site due to missing data, no confidence in site safety, or both. In this case, the multicopter executes a coverage path to explore the area and evaluate each overflown area to find a safe landing site. These three fail-safe algorithms integrate map generation, flight planning, and area coverage capabilities. Simulations demonstrate proposed protocols against "low battery" anomalies that might be encountered during low-altitude urban multicopter missions.

The remainder of the chapter is organized as follows. Sec. 2.2 summarizes the fail-safe landing problem and the three alternative protocols proposed in this dissertation. Sec. 2.3 describes the PLUTO and MapPLUTO databases and summarizes the processing required to prepare map data for simulation. Sec. 2.4 provides a more detailed description of the planning algorithms used for the three fail-safe protocols. Sec. 2.5 presents results from simulation case studies in which each of the three

protocols is applied to a multicopter experiencing a fail-safe trigger while flying over Midtown Manhattan. Finally, conclusions and future work are discussed in Sec. 2.6.

2.2 Problem Statement

Three alternative fail-safe protocols are examined in this dissertation for urban multicopter flight. The decision regarding which protocol to execute depends on the amount of information available when the fail-safe protocol is invoked. This work assumes the UAS can localize and navigate safely, despite its anomaly. Table 2.1 summarizes protocol selection as a function of available information.

Protocol	Cleared-to-Land	Land Use	Building: Location	Shape	Height
1	✓	N/A	✓	✓	✓
2	-	✓	✓	✓	✓
3	-	-	✓	✓	✓

Table 2.1: Alternative fail-safe multicopter planning protocol selection as a function of available (✓) versus unavailable (-) information.

In the first protocol, the multicopter determines the closest rooftop flagged *Cleared-to-Land*. A flagged rooftop is assumed free of all obstacles and people; this flag is only set when the roof has a geometry that supports a safe multicopter landing. A* motion planning is used to generate a flight plan to the selected rooftop.

In the second protocol, the multicopter first finds a reachable landing site using the *Land Use* categories available for each tax lot. As the multicopter travels along its trajectory, it observes overflown rooftops for landing site alternatives. If the multicopter discovers a viable landing rooftop en route, the multicopter will execute an immediate landing protocol to eliminate the risk posed by continuing the flight to the more distant site. A* motion planning is again applied.

In the third protocol, the UAS does not have sufficient information to select a landing site beyond the onboard sensors' field of view. The multicopter generates

a spanning tree coverage (STC) [63] flight plan in search of a viable rooftop for a fail-safe landing as a last resort. Each protocol is further defined in detail in Sec. 2.4.

2.3 UAS Planner Database Generation

New York City (NYC) maintains the PLUTO and MapPLUTO public databases as part of their Open Data initiative [62]. These databases contain tax lot, land use, and geographic data for the five boroughs of NYC. We consider only Manhattan in this dissertation. Table 2.2 describes the relevant content in PLUTO and MapPLUTO.

Feature	Description	Source
ShapeX	X-coordinates of tax lot polygon.	MapPLUTO
ShapeY	Y-coordinates of tax lot polygon.	MapPLUTO
Block	Tax block number in which the tax lot resides.	PLUTO
Lot	Tax lot number unique within each tax block.	PLUTO
BldgClass	Primary use of structures, facilities, and tax lot area.	PLUTO
NumFloors	Number of stories of the tallest tax lot building.	PLUTO

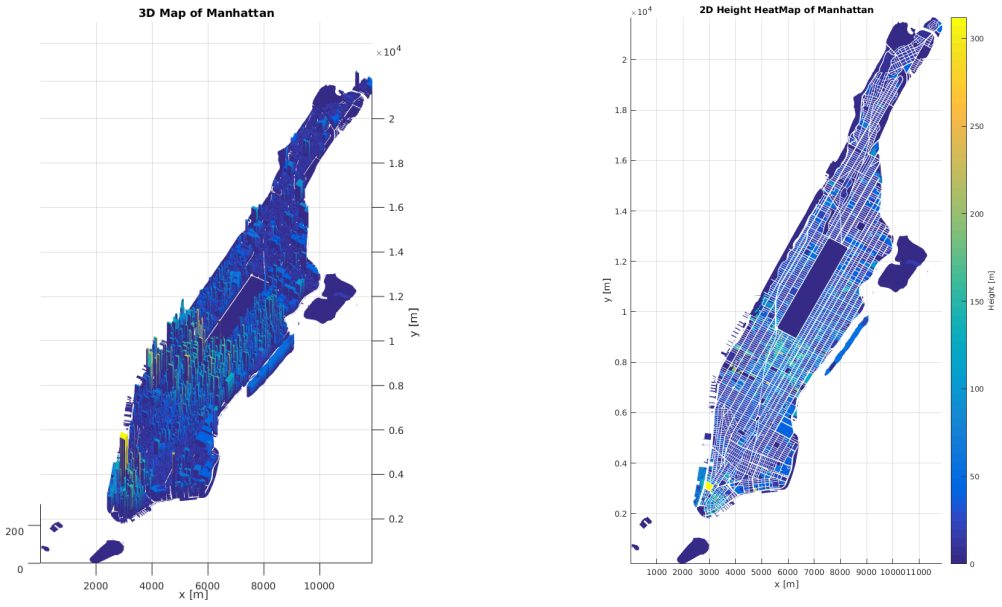
Table 2.2: Raw geospatial tax lot data supplied by PLUTO and MapPLUTO.

For our database, each tax lot was assigned a unique identifier consisting of a block number followed by a lot number, *Block-Lot*. Buildings in a tax lot were approximated by extruding a single building per tax lot with the building's geometry consistent with that of the tax lot itself, *ShapeX* and *ShapeY*. The number of building floors, *NumFloors*, at 10 feet per floor was used to estimate building height. Lastly, land use data was derived from *BldgClass* and condensed into two main categories:

- Buildings: Residences, commerical spaces, factories, hotels, stores, offices, etc.
- Outdoor Spaces: Parks, playgrounds, gardens, recreational facilites, etc.

In this chapter, no distinction was made for *restricted sites*, e.g., government facilities, churches, public monuments, and schools. Fig. 2.3 shows 2D and 3D representations of all buildings mapped in this chapter. All position data were converted to

a local UTM 18N (EPSG:32618) coordinate reference system (CRS). The Universal Transverse Mercator (UTM) coordinate projection allows metric calculations directly defining axes (easting, northing) in meters.



(a) 3D PLUTO and MapPLUTO map. (b) 2D heat map of building heights.

Figure 2.3: Building extrusion/height maps of the Manhattan borough of New York City derived from PLUTO and MapPLUTO databases used for UAS planning and simulation.

2.4 Flight Planning Protocols

Flight planning prescribes a trajectory or waypoint sequence from an initial state to a final destination. Battery or fuel level and local map information are influential. In-flight replanning must occur with minimal delay to support urgent landings given an emergency or anomaly. Three urgent landing protocols are summarized below.

2.4.1 Protocol 1: Cleared to Land

The *Cleared-to-Land* Manhattan map in Fig. 2.4 highlights buildings with rooftops that can support urgent multicopter landings. It assumed that if a building rooftop

has been flagged, its geometry guarantees a safe landing and there exist no obstacles to circumvent. In this study, each building was randomly assigned a *Cleared-to-Land* flag prior to simulation.

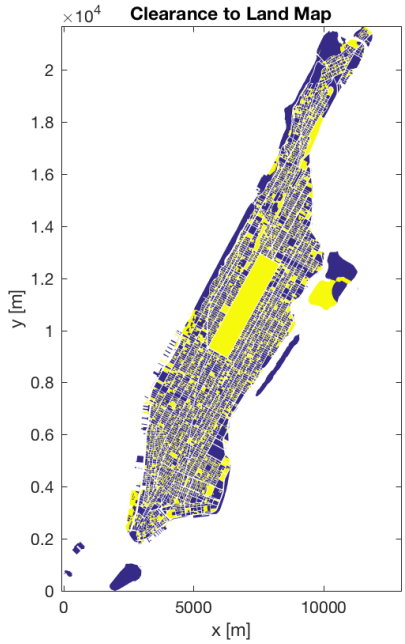


Figure 2.4: Unobstructed safe multicopter landing sites depicted in yellow.

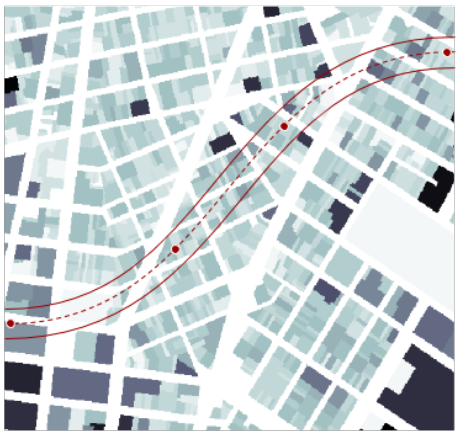
The planner searches for the nearest *Cleared-to-Land* building with a height shorter than the multicopter's initial altitude by at least two meters using A* over a 2m x 2m Manhattan grid. A 10 m radius linear decaying repulsive field is wrapped around building perimeters to ensure path clearance from each building for planning. A distance cost function is assumed along with a Euclidean distance admissible heuristic.

2.4.2 Protocol 2: Observation Tube

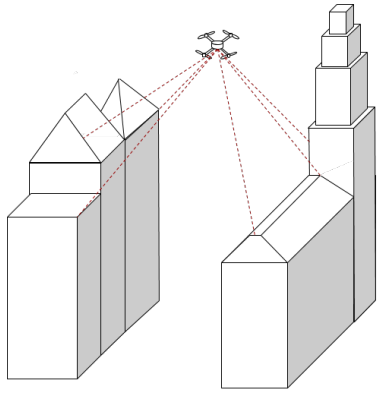
The second fail-safe protocol determines a path to a tentative landing site per Protocol 1 but also seeks opportunities to land sooner. The multicopter references land use information to designate an initial landing site, likely an open area within range (i.e., within the multicopter's landing footprint). While following the tentative

landing plan, imagery collected en route generates an observation "tube" of data searched for closer (flat rooftop) alternate landing sites per Fig. 2.5.

The multicopter must fly sufficiently near the building to characterize rooftop geometry and determine whether a building is suitable for landing. To account for different roof types, each building is assigned a "Rooftop Landing Probability" in the simulated city map over a uniform distribution. If the examined rooftop meets a predetermined probability threshold based on safety/confidence, the multicopter will divert to land on the closer safe rooftop.



(a) Imager observation tube along a tentative landing path.



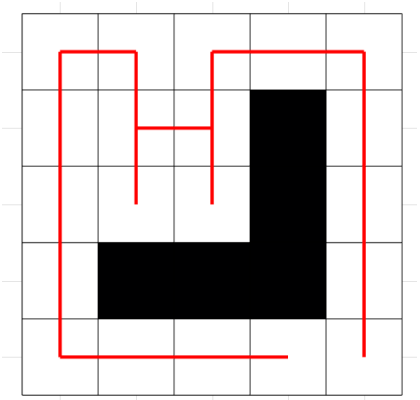
(b) Multicopter scanning nearby rooftops to find an alternative landing site.

Figure 2.5: Observation tube opportunistic alternate landing site protocol.

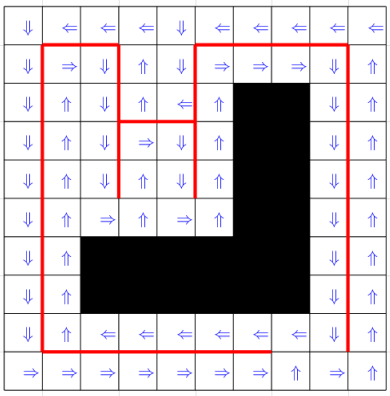
2.4.3 Protocol 3: Spanning Tree Coverage

In this final case study, the multicopter does not assume knowledge of clear, flat rooftop sites but instead follows a spanning tree coverage (STC) [63] path assuming known building location/height data that allows an onboard imager to identify a nearby landing site. With STC, the search area is gridded with cells of side length $2\delta_{view}$; the value of δ_{view} depends on sensors' field of view or range. A graph with unobstructed adjacent cell edges is used to construct a spanning tree from the start grid without cycles using Prim's algorithm [64] per Fig. 2.6. All grid cell centers are

treated as nodes of an undirected graph with edges formed between adjacent cells in obstacle-free areas, where buildings are shorter than UAS altitude during planning.



(a) Imager observation tube along a tentative landing path.



(b) Multicopter scanning nearby rooftops to find an alternative landing site.

Figure 2.6: Observation tube opportunistic alternate landing site protocol.

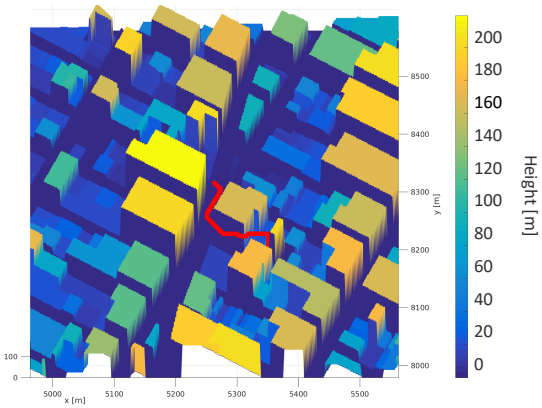
A coverage flight path circumnavigates the spanning-tree counterclockwise (or clockwise), returning to the start node if the multicopter's energy level allows. Traversing the STC path does not guarantee the multicopter will identify a safe rooftop landing site, so Protocol 3 would only be preferred if known safe sites were not reachable.

2.5 Results

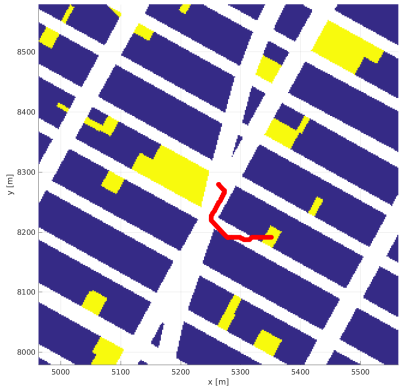
Case studies consider a small "media drone" with abnormal battery behavior, e.g., unexpected low energy, triggering a fail-safe protocol. The multicopter does not have sufficient energy to return to its designated home point when given a densely populated overflight area, and it must select an alternative landing site beyond the onboard sensor field of view. Results from each fail-safe protocol are presented below for a case in which the multicopter is initially at an above ground level (AGL) height of 100m over the center of a densely-occupied Times Square.

2.5.1 Case Study 1: Cleared to Land

In this case, available rooftop landing sites are mapped. The UAS selects a nearby retail store's rooftop flagged as *Cleared-to-Land*. A flight plan generated with A*, as detailed in Sec. 2.4, is followed to the selected rooftop per Fig. 2.7.



(a) 3D landing flight path from Times Square to a nearby rooftop.



(b) Overhead of landing path; yellow sites are *Cleared-to-Land* rooftops.

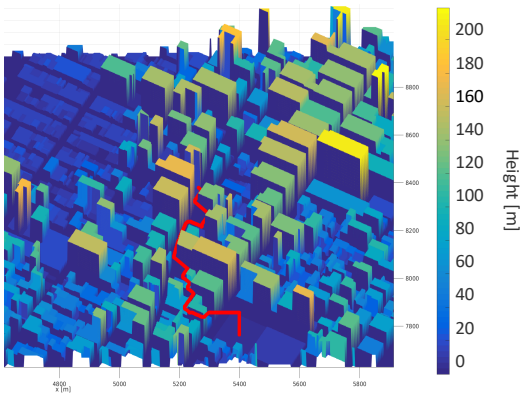
Figure 2.7: Multicopter traveling to the nearest rooftop designated *Cleared-to-Land*.

2.5.2 Case Study 2: Observation Tube

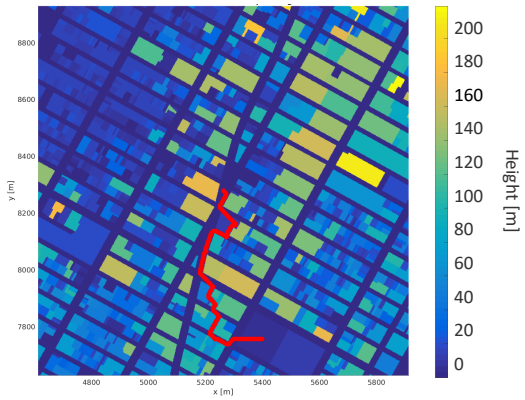
In this case, the multicopter does not have access to *Cleared-to-Land* sites but uses map data to plan a tentative landing, e.g., to the nearest park, using map-based A*. For this case simulation, Bryant Park in Midtown Manhattan was selected as a tentative landing site.

During flight, observed rooftops and open ground can be marked safe for landing, given a confidence level greater than a threshold. If the threshold is high, the multicopter may pass potential landing sites. With a high probability measure (0.95), Fig. 2.8 shows the multicopter flying to Bryant Park. In contrast, Fig. 2.9 shows diversion to an alternative landing site with a lower threshold (0.85). For this case, the multicopter discovers a viable landing rooftop en route, a nearby parking structure with an unoccupied roof. This case study illustrates how real-time fusion of map and

sensor data can better inform urgent landing.

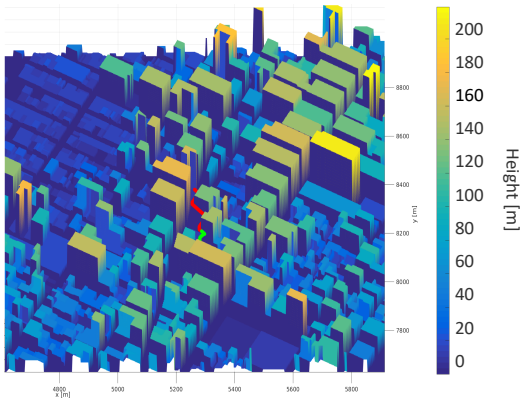


(a) 3D view of multicopter traveling to the planned Bryant Park landing site.

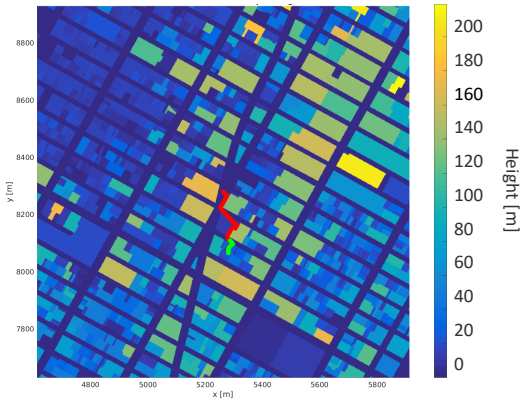


(b) Overhead view of the multicopter traveling to Bryant Park.

Figure 2.8: Multicopter urgent landing plan to Bryant Park using Protocol 2.



(a) 3D view of rooftop landing path.



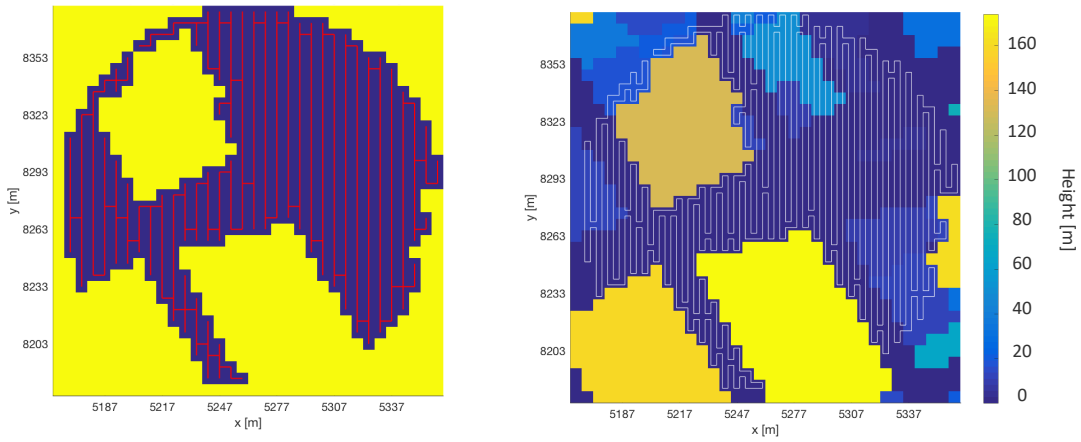
(b) Overhead of rooftop landing path.

Figure 2.9: Landing path for a multicopter diverting to a safe rooftop landing site discovered along its Observation Tube (Protocol 2) flight to Bryant Park.

2.5.3 Case Study 3: STC

For this last case study, the multicopter has no landing site database. The multicopter uses STC to define a coverage path over a specified search radius, 100m for this study. Typically, the search radius would be related to the expected remaining UAS range. Fig. 2.10 shows the generated spanning tree and STC plan. While

no ideal rooftop might be detected, the acceptability threshold can be adjusted over time to select either the first fully viable rooftop or the best rooftop observed within a prescribed energy margin constraint or deadline.



(a) Spanning tree (Prim's algorithm) (b) STC flight plan

Figure 2.10: STC constant altitude flight over 100m radius search near Times Square.

2.6 Conclusion

The above case studies illustrate how the three complementary urgent landing protocols would locally guide a small UAS to a safe landing given different local landing site data sets. For this work, we adopted specific existing flight planning algorithms relevant for the immediate and cluttered area around Times Square, but that would face computational overhead challenges given a much larger multicopter range to landing. This result motivated additional research in motion planning algorithm selection as described in subsequent chapters.

In the future, this work can be extended to account for localization uncertainty, given the inaccuracies of GPS in an urban canyon by using state estimation techniques. Simultaneous localization and mapping (SLAM) techniques and online STC-based flight planning could be used when flying in unfamiliar urban areas, motivating online versions of the proposed protocols. Finally, given access to real-time data

sources such as cell-phone call detail reports [10], proposed methods could be extended to include viable ground landing sites in areas with low population densities or data-based occupancy estimates.

CHAPTER III

Algorithm Selection Maps and Metrics

3.1 Introduction

The algorithm selection problem (ASP) requires qualitative and quantitative metrics to inform the ASP of user/agent, algorithm, and configuration space preferences and constraints. Algorithm metrics can be defined from learned performance models [65, 66], statistical measures [67], abstract features [68], and classical definitions [69]. Additional metrics account for application-specific considerations. Table 3.1 summarizes classical algorithm properties relevant when selecting a motion planner.

Property	Description
Completeness	A solution is returned if one exists; otherwise, failure is returned.
Soundness	If a solution is returned, it is feasible.
Complexity	Memory usage and/or execution time measured with theoretical upper bounds and/or large-scale Monte Carlo simulation.
Kinodynamics	Planning solutions are consistent with UAS kinematics and dynamics properties and constraints.
Dynamic Env.	Handling of environment (env.) properties change over time.
Uncertainty	Planner accounts for uncertainty in robot or environment states.
Optimality	A best solution is returned with respect to a given metric or combination of metrics.

Table 3.1: Classical algorithm properties relevant to motion planning.

For aerospace applications, completeness, soundness, and manageable computational complexities are desired to support safe real-time decisions. Kinodynamics,

environment properties, and state uncertainties are typically managed through planner costs and constraints. Fixed-wing aircraft optimize flight altitude (atmospheric density), velocity, climb rate, lift/drag ratio [70], range or endurance [71], and hazardous weather avoidance [72]. Multicopter UAS operating at specified low altitudes optimize over separation from obstacles, energy, and mission time [2]. Communication [73] and navigation [74] metrics are key considerations where line-of-sight signals may be blocked. A Pareto front analysis offers insight into balancing competing metrics [75, 76, 77].

This chapter explores motion planning metrics targeted for urban multicopter flight. We define a set of map-based m_m, path-based m_p, and software execution-based metrics m_s to inform algorithm selection. Map-based metrics m_m describe the operating environment by constructing a collection of GPS/Lidar navigation performance, population density, and obstacle risk exposure metric maps. Path-based metrics m_p account for a vehicle's energy consumption and distance traveled along a path under construction. Software-based metrics m_s measure memory consumption and execution time of an algorithm. This chapter focuses on a detailed analysis of map-based metrics with path-based and software-based metrics utilized in the Monte Carlo and ASP studies described in Ch. IV.

Map-based metrics are derived offline from open-source geospatial, satellite, and census databases. Using cloud computing, each database is processed and transformed to generate a collection of discretized metric maps representative of the borough of Manhattan in New York City. Small UAS maps were generated for low, medium, high, and flight ceiling altitudes over daytime and nighttime periods to examine the different metric effects across the urban canyon. GPS navigation, lidar visibility, and risk exposure were most affected by flight altitude, with population cost metric set as a function of commercial and residential regions plus time-of-day.

The remainder of the chapter is organized as follows. Sec. 3.2 summarizes pro-

posed planning metrics and their use. Sec. 3.3 defines map-based, path-based, and sotware-based metrics in detail followed by a description of map-based metric specifications as discretized feature maps in Sec. 3.4. Map-based metric results are presented in Sec. 3.5 followed by conclusions and future work in Sec. 3.6.

3.2 Problem Statement

We define map-based metrics m_m, path-based metrics m_p, and software execution-based metrics m_s to inform sUAS flight planners and the algorithm selection problem (ASP) per Table 3.2. All metrics are subsequently defined with an in depth analysis of all map-based metrics. Path-based and software-based metrics are presented in Chap. IV when solving the ASP.

Table 3.2: Motion planning ASP metrics classified by type.

Metric	Description	Type
m_{gps}	GPS pseudorange position uncertainty	map
m_{lidar}	Lidar-based local map uncertainty	map
m_{obs}	Obstacle occurpancy	map
m_{pop}	Overflown population estimate	map
m_{risk}	Proximity to obstacles enroute	map
m_{dist}	Distance traveled along a path	path
m_{time}	Algorithm execution time	software
m_{mem}	Algorithm memory usage	software

For a given operating bounding box $\mathcal{L}$, a collection of metric maps $\mathcal{H}$ is generated describing all m_m. $\mathcal{H}$ is generated by explicitly calculating each metric at every point characterized by a Cartesian grid over $\mathcal{L}$ with resolution δ_{res}. Building outlines are extracted from OpenStreetMap [78] and city database building height information to estimate lidar visiblity and obstacles/risk exposure in the area. CelesTrak [79] and Skyfield [80] simulate GPS operational satellites above $\mathcal{L}$ to estimate GPS-based positioning accuracy. Census data [81] is used to estimate population patterns accross $\mathcal{L}$. In combination with online path-based metric m_p a weighted cost optimization is

performed to compute an obstacle-free path to a targeted landing site as shown in Fig. 3.1.

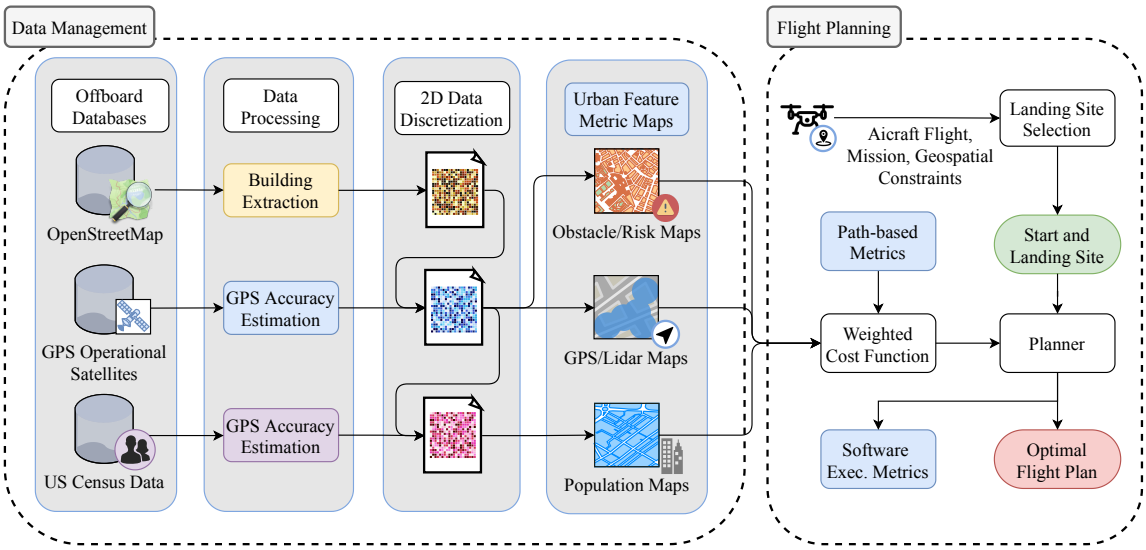


Figure 3.1: Data flow for map-based metric generation in data-driven multicopter flight planning.

3.3 Metric Definitions

3.3.1 GPS Uncertainty

GPS receivers communicate with a global navigation satellite system (GNSS) to estimate their geographical location using trilateration. Given receiver/satellite pair, a pseudorange measurement is estimated as [82]:

$$\hat{\rho}_{rc,sat} = \rho_{rc} + c(t_{sat} - t_{rc}) + \epsilon \tag{3.1}$$

where ρ_{rc} is receiver range, c is the speed of light, t_{sat} and t_{rc} are the satellite/receiver clock readings, and ϵ_{UERE} captures any User Equivalent Range Errors (UEREs), e.g., atmospheric, clock, signal, and multipath errors.

Geometric dilution of precision ($GDOP$) describes error propagation from satellite geometry: dispersed satellites reduce uncertainty while clustered satellites increase it

[83]. $GDOP$ can be expressed as:

$$GDOP(x, y, z, t) = \sqrt{PDOP(x, y, z)^2 + TDOP(t)^2} \tag{3.2}$$

where $PDOP$ and $TDOP$ are position/time dilutions of precision, respectively. DOP values between 1 to 20 [4] quantify GPS reliability as summarized in Table 3.3.

DOP	Rating	Description
1	Ideal	Highest precision possible.
$1-4$	Excellent	Measurements are considered accurate except for the most sensitive applications.
$4-6$	Good	Represents the minimum acceptable loss in accuracy.
$6-8$	Moderate	May still be used but only recommended in obstacle free environments.
$8-20$	Fair	Readings should be dismissed or only serve to compute a rough estimate.
>20	Poor	Unreliable and should not be used.

Table 3.3: DOP Value Rating [4].

For n visible satellites, pseudo ranges offer a fast approximation of $PDOP$. Applying a first-order Taylor expansion to the true range, pseudorange to the ith satellite is computed as:

$$\hat{\rho}_{rc,i} = \frac{x_{rc} - x_{sat,i}}{r_i} x_{rc} + \frac{y_{rc} - y_{sat,i}}{r_i} y_{rc} + \frac{z_{rc} - z_{sat,i}}{r_i} z_{rc} + c(t_{sat,i} - t_{rc}) \tag{3.3}$$

where x_{rc}, y_{rc}, z_{rc}, t_{rc} and $x_{sat,i}$, $y_{sat,i}$, $z_{sat,i}$, $t_{sat,i}$ are the positions/clock readings of the receiver and ith satellite respectively. This information can be expressed as a

linear system G_{gps} and state vector $\boldsymbol{\chi}_{gps}$:

$$G_{gps} = \begin{pmatrix} \frac{x-x_1}{r_1} & \frac{y-y_1}{r_1} & \frac{z-z_1}{r_1} & -1 \\ \frac{x-x_2}{r_2} & \frac{y-y_2}{r_2} & \frac{z-z_2}{r_2} & -1 \\ \vdots & \vdots & \vdots & \vdots \\ \frac{x-x_n}{r_n} & \frac{y-y_n}{r_n} & \frac{z-z_n}{r_n} & -1 \end{pmatrix} \qquad \boldsymbol{\chi}_{gps} = \begin{pmatrix} x \\ y \\ z \\ c \cdot t_{rc} \end{pmatrix} \tag{3.4}$$

with a best linear unbiased estimator (BEST), covariance $\Sigma_{gps} = \left(G_{gps}^T G_{gps}\right)^{-1}$ and dilutions of precision defined per: [84]:

$$PDOP = \sqrt{\Sigma_{11,gps}^2 + \Sigma_{22,gps}^2 + \Sigma_{33,gps}^2}, \quad TDOP = \sqrt{\Sigma_{44,gps}^2} \tag{3.5}$$

Accounting for visible satelllites, we define a motion planning GPS map-based uncertainty metric m_{gps} or cost c_{gps} as:

$$m_{gps}(x, y, z, t) = \frac{GDOP_{thresh} - \min(GDOP(x, y, z, t), GDOP_{cut})}{GDOP_{thresh} - 1} \tag{3.6}$$

$$c_{gps}(x, y, z) = 1 - m_{gps}(x, y, z) \tag{3.7}$$

where $GDOP_{thresh}$ is a worst-case cutoff value for safe flight.

3.3.2 Lidar Visibility

Lidar provides a local obstacle point cloud to assure safe navigation through complex spaces and support local-area mapping. In GPS-denied areas, Lidar [85] can be used for inertial navigation by tracking mapped buildings and other landmarks. Lidar uses a laser's reflection time to estimate distances to objects. Lidar can be configured as a dome or cylindrical puck for local and longer-range applications.

The puck configuration modeled in this work uses b_{lidar} equiangular beams that revolve to scan at f_{lidar} equiangular positions capturing $n_{lidar} = b_{lidar} \cdot f_{lidar}$ points

per revolution. Because $f_{lidar} >> 1$, n_{lidar} is impractical for metric normalization, so we propose number of returned scan readings (where an obstacle is within lidar range) as a lidar metric. A scan reading is recorded if any beam of the jth scan, $j = 1, 2, \cdots, f_{lidar}$, intersects an obstacle in Ω_{obs} within range r_{lidar} from the sUAS:

$$scan(j) = \begin{cases} 1, & \text{if } \exists i \text{ s.t. } \overleftrightarrow{b_{\mathcal{O}} b_{i,j}} \cap \Omega_{obs} \neq \emptyset \\ 0, & \text{otherwise} \end{cases} \tag{3.8}$$

where $i \in \{1, 2, \ldots, b_{lidar}\}$, $b_{\mathcal{O}}$ is the origin point of all beams, and $b_{i,j}$ is the ith lidar beam point for the jth scan a distance r_{lidar} away with an elevation angle β_i.

A count of total scan returns $s_{lidar}(x, y, z, r_{lidar}) = \sum_1^{f_{lidar}} scan(j)$ is then compared to the total number of possible scan returns in the following lidar metric m_{lidar} or cost c_{lidar}:

$$m_{lidar}(x, y, z, r_{lidar}) = \frac{s_{lidar}(x, y, z, r_{lidar})}{f_{lidar}} \tag{3.9}$$

$$c_{lidar}(x, y, z, r_{lidar}) = 1 - m_{lidar}(x, y, z, r_{lidar}) \tag{3.10}$$

3.3.3 Obstacle Occupancy

Obstacle maps allow motion planners to define free $\mathcal{C}_{free}$ and obstacle $\mathcal{C}_{obs}$ configuration spaces. We define an obstacle occupancy metric m_{obs} to penalize flight paths with points that intersect obstacles such that:

$$m_{obs}(x, y, z) = c_{obs}(x, y, z) = \begin{cases} 0, & \text{if } (x, y, z) \cap \mathcal{C}_{obs} = \emptyset \\ 1, & \text{otherwise} \end{cases} \tag{3.11}$$

3.3.4 Population Density

Flying low imposes a nontrivial risk to the overflown population. Population metric m_{pop} estimates expected normalized population density for each weekday. Population can be estimated from census data or dynamic sources such as mobile phone activity [10]. For Manhattan, turnstile and taxi data have also been used to estimate population [86]. Similar information is not available across multiple cities, so we propose extrapolating population estimates directly from census data.

Manhattan's overall population has been estimated as follows [5]:

Table 3.4: Dynamic population estimates in millions for Manhattan in 2010. [5].

	Work Week	Weekend
Daytime	3.94	2.90
Nighttime	2.05	2.05

A city's population varies throughout the day. Due to typical work hours, e.g., 9-to-5, population estimates in *commercial* areas are higher during the day. As people return home after work *residential* areas become densely populated during the evening. To estimate occupancy for each census map grid, we assume census data pop_{census} for nighttime population and scale daytime population by a factor Γ determined based on area zoning (commercial or residential) such that:

$$\hat{pop}(k, l, \Gamma) = \begin{cases} \Gamma \cdot pop_{census}[k] & \text{if } l = day \\ pop_{census}[k] & \text{if } l = night \end{cases} \tag{3.12}$$

where l denotes time of day and k is the census grid index.

The following population density metric is then defined:

$$m_{pop}(x, y, l, \Gamma) = \frac{\hat{pop}(k(x, y), l, \Gamma)}{\hat{pop}_{norm}(\mathcal{B}, l)} \qquad c_{pop}(x, y) = m_{pop} \tag{3.13}$$

where $\hat{pop}_{norm}(\cdot)$ is maximum daytime or nighttime population density over bounding region $\mathcal{B}$ and $k(\cdot)$ is an indexing function relating census index to world coordinates.

3.3.5 Risk Proximity Metric

Numerous safety risks can be defined, but in this work, we focus on one particular risk: proximity to nearby buildings or terrain defined as threshold-based rectifier function. The building map is used to compute the distance to the closest obstacle surface, $d_{close}(x, y, z)$, for each map grid or point in space. For a specified distance threshold, d_{thresh}, a proximity risk is defined as:

$$m_{risk}(x, y, z) = \min\left(\frac{d_{close}}{d_{thresh}}, 1\right) \qquad c_{risk}(x, y, z) = 1 - m_{risk}(x, y, z) \tag{3.14}$$

3.3.6 Distance Path Metric

The expected distance traversed is given by:

$$m_{dist}(t_0, t_f) = \int_{t_0}^{t_f} |v(t)| dt \tag{3.15}$$

where t_0 and t_f are initial and final planned flight times and $v(\cdot)$ is velocity magnitude. This function can also be written as a summation of N segment lengths of path $\boldsymbol{\zeta}$:

$$m_{dist}(\zeta, N) = \sum_{i=1}^{N} \sqrt{(\zeta_{x,i} - \zeta_{x,i-1})^2 + (\zeta_{y,i} - \zeta_{y,i-1})^2 + (\zeta_{z,i} - \zeta_{z,i-1})^2} \tag{3.16}$$

3.3.7 Software-based Metrics

Execution time m_{time} and memory usage m_{mem} metrics are given by:

$$m_{time} = t_{a,f} - t_{a,0} \qquad m_{mem} = \max(\Pi(a, t_{a,0}, t_{a,f})) \tag{3.17}$$

where $t_{a,0}$ and $t_{a,f}$ denote start and end times for executing algorithm a and $\Pi(\cdot)$ records the computer memory usage throughout the algorithm execution.

3.4 Map Generation Overview

Each Cartesian metric map of specified resolution defines a metric value for each grid. For this investigation, metric maps cover an area $\mathcal{L}$ with a width 10km and height of 20km centered in Manhattan per Fig. 3.2. Maps with 2m, 5m, and 10m

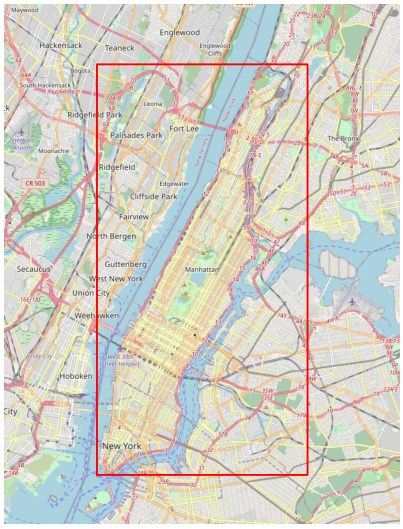


Figure 3.2: Planning configuration space area $\mathcal{L}$ for Manhattan case studies.

resolution were defined. The 2m value coincides with current small UAS positioning and obstacle avoidance capabilities. Height-dependent metrics were computed for UAS altitudes of 20m, 60m, 122m (FAA maximum altitude for sUAS), and 600m AGL (above ground level), capturing low, medium, high, and ceiling-altitude flight.

3.4.1 Obstacle Maps

OpenStreetMap (OSM) [78] data was processed to extract a building-based obstacle map $\mathcal{H}_{obs}$ from ways and relations using attribute labels. OSM data was translated from its WGS84 CRS to UTM 18N as in Chap. II. Extracted polygons Ω_{obs} were rasterized per map resolution. The height of the kth extracted polygon z_k located at

grid point (x, y) was compared to a given altitude z such that:

$$\mathcal{H}_{obs}(x, y, z) = \begin{cases} 1 & \text{if } z_k \geq z \\ 0 & \text{otherwise} \end{cases} \tag{3.18}$$

3.4.2 GPS Maps

GPS metric maps $\mathcal{H}_{gps}$ describe expected GPS accuracy for the Manhattan urban canyon. For a given grid point (x, y, z) and date/time information $\mathcal{D}$, positions of overhead satellites are predicted using CelesTrak [79] and Skyfield [80]. Rays are cast to above-horizon satellites and checked for collisions against extruded buildings in Ω_{obs}. With less than four visible satellites ($N_{sats} < 4$), m_{gps} is set to zero; otherwise the GPS pseudorange and covariance matrices are used to calculate m_{gps}:

$$\mathcal{H}_{gps}(x, y, z) = \begin{cases} m_{gps}(x, y, z) & \text{if } N_{sats} \geq 4 \\ 0 & \text{otherwise} \end{cases} \tag{3.19}$$

3.4.3 Lidar Maps

Lidar metric maps $\mathcal{H}_{lidar}$ estimate metric m_{lidar}, the expected percentage of lidar range returns. It is assumed that the vehicle is equipped with a b-beam lidar configured in a parallel configuration, i.e., the aircraft's z_{body} and lidar's rotation axis are colinear. The beams share an equiangular spread along the elevation angular range of $[-\beta, \beta]$. In this configuration, only the $\frac{b}{2}$ beams in the range $[-\beta, -\frac{\beta}{b/2}]$ are eligible for calculating $scan(\cdot)$ as the remainder are blocked by the vehicle. Hence, the ratio of scan returns per revolution at each grid point (x, y, z):

$$\mathcal{H}_{lidar}(x, y, z, r_{lidar}) = m_{lidar}(x, y, z, r_{lidar}) \tag{3.20}$$

3.4.4 Population Maps

Population metric maps $\mathcal{H}_{pop}$ are computed based on zoning and census data compiled into the normalized population metric m_{pop}. Census values are adjusted by Γ_{comm} or Γ_{resi} during the day ($l = day$) for commercial and residential areas to account for commuting patterns.

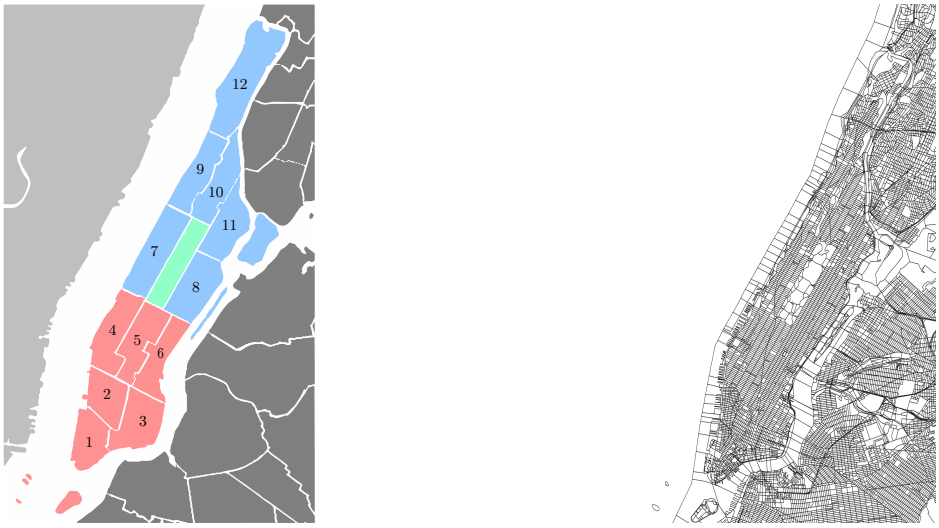
Manhattan is divided into twelve districts, starting at its southernmost neighborhood, i.e., the Financial District, to its northernmost neighborhood, i.e., Harlem, as shown in Fig. 3.3. The lower districts (1-6) are composed of businesses, government buildings, and tourist attractions. In contrast, the upper districts (7-12) consist mostly of single and family residences. Defined by NYC Department of City Planning [87], in this investigation, the twelve districts are labeled as shown on Table 3.5.

Table 3.5: Manhattan districts with their important neighborhoods' labeled type.

Number	Neighborhoods	Type
01	Financial District, Civic Center	Commercial
02	West Village, Greenwich Village, Soho	Commericial
03	Chinatown, East Village, Noho	Commericial
04	Chelsea, Clinton, Hell's Kitchen	Commericial
05	Union Square, Madison Square, Times Square	Commericial
06	Gramercy, Murray Hill, Turtle Bay	Commericial
07	Lincoln Square, Upper West Side, Manhattan Valley	Residential
08	Lenox Hill, Upper East Side, Yorkville	Residential
09	Morningside Heights, Hamilton Heights	Residential
10	Central Harlem	Residential
11	East Harlem	Residential
12	Inwood, Washington Heights	Residential

Population data is derived from the 2010 United States Census [81]. The WGS84 census block polygons represent the smallest geographic unit used by the US Census Bureau to estimate the number of residents in a block. Each census block entry includes a cumulative population count for that block and is assigned a district number 1-12 if the census block and district outline fully intersect. Any census block

overlapping multiple outlines is assigned the district polygon's label with the largest intersection by area. Any census block within $\mathcal{L}$ but not in Manhattan, i.e., the Bronx or Queens, is given a district label of 13 and labeled as residential. All data geospatial data is converted to UTM 18N for consistency.



(a) Figure of manhttan districts (b) Figure of mahattan census polygons.

Figure 3.3: Manhattan community districts and census blocks.

Two metric population maps are created for daytime and nighttime hours per altitude, resolution pair. For daytime hours, blocks in commercial areas are scaled by a Γ_{comm} modifier and Γ_{resi} for residential blocks to meet the daytime population constraint of approximately 4 million [5]. No modifiers are applied for nighttime operations.

Accounting for block type and time-of-day l, the population map $\mathcal{H}_{pop,l}$ has the following form:

$$\mathcal{H}_{pop,l}(x,y) = \begin{cases} m_{pop}(x,y,l,\Gamma_{comm}) & \text{if } label[k] = \text{commericial} \\ m_{pop}(x,y,l,\Gamma_{resi}) & \text{if } label[k] = \text{residential} \end{cases} \tag{3.21}$$

where $k = k(x,y)$ returns census block index k containing grid point (x,y).

3.4.5 Risk Maps

The final metric map set $\mathcal{H}_{risk}$ quantifies building obstacle risks in the urban canyon as a function of the proximity risk metric m_{risk}:

$$\mathcal{H}_{risk}(x, y, z) = m_{risk}(x, y, z) \tag{3.22}$$

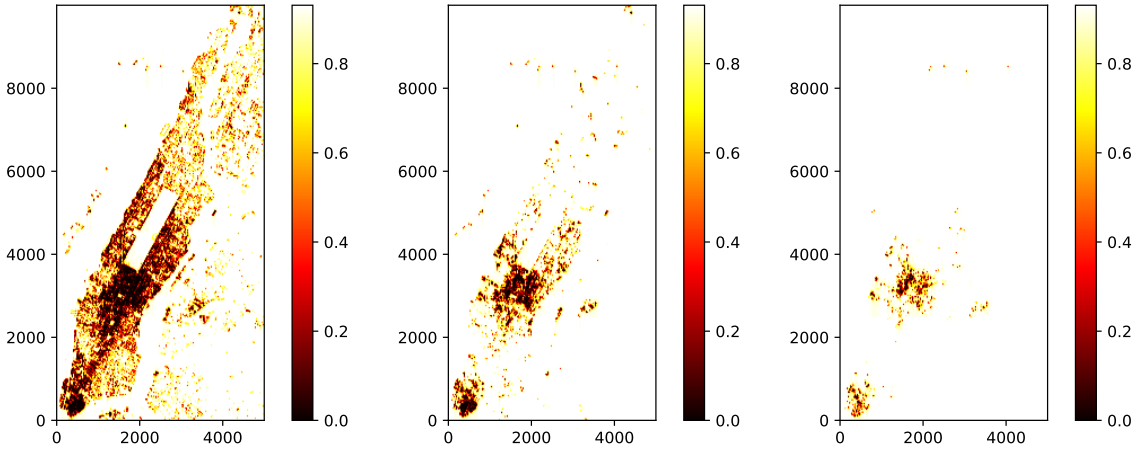
Note that cost map equivalents for each metric map can be computed by following the metric-to-cost conversions presented in Sec. 3.3.

3.5 Manhattan Metric Map Results

Metric maps over Manhattan region $\mathcal{L}$ at three different resolutions (2m, 5m, and 10m) were generated for four small UAS AGL flight altitudes: 20m (low-altitude), 60m (medium-altitude), 122m (high-altitude), and 600m (ceiling-altitude). These are our labelings that offer sUAS flight paths that range from deep inside the NYC urban canyon (low-altitude) to above all buildings (ceiling-altitude). Specific altitude ranges defining UAS *low-altitude*, *medium-altitude*, and *high-altitude* flight vary across the literature.

Fig. 3.4 shows GPS maps for low, medium, and high-altitude flight. GPS metric scores are normalized between 0 and 1, where $m_{gps} = 1$ indicates the highest accuracy. As expected, GPS accuracy is highest in building-free areas, i.e., the Hudson River or Central Park, or small-building areas, i.e., New Jersey. GPS accuracy decreases in low-altitude urban canyon regions with skyscrapers and other tall buildings.

For medium-altitude flight, the effects of urban canyon flight lessen. Upper Manhattan and Brooklyn (lower right) are now areas with high GPS accuracy. Similarly, high GPS accuracy areas now appear in Lower Manhattan but to a lesser extent. The Financial District (bottom left) and Midtown Manhattan (below Central Park) still



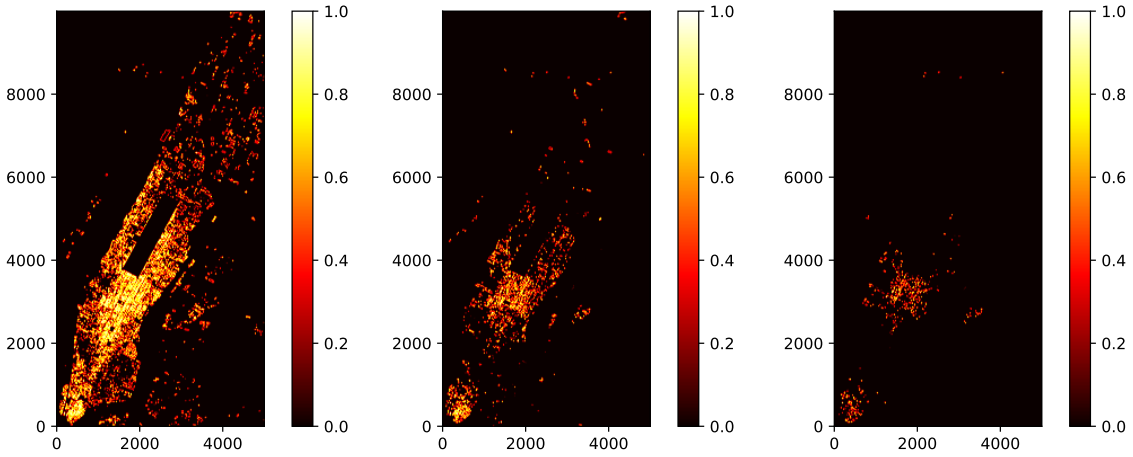
(a) GPS 2m res map at 20m. (b) GPS 2m res map at 60m. (c) GPS 2m res map at 122m.

Figure 3.4: GPS metric maps for low, medium, and high-altitude urban flight.

include low GPS accuracy regions. This is to be expected as these areas are known for their tall buildings, e.g., One World Trade Center and Central Park Tower. The UAS primarily operates above the urban canyon at high and ceiling flight altitudes with near-perfect GPS accuracy.

Fig. 3.5 shows expected Lidar performance for low-altitude and medium-altitude flight. In contrast to GPS, Lidar performance is better at lower altitudes since the urban canyon offers in-range point cloud data and better visibility of its surroundings. In low-altitude flight, Lidar performance is highest in the East Side, West Side, Midtown, and Downtown Manhattan areas densely packed with commercial and tourist high-rises. Weak Lidar returns can be found in Uptown Manhattan, New Jersey, Brooklyn, and Queens, areas with mostly low-rise and residential buildings.

Medium-altitude Lidar analysis shows a significant drop in performance. Of the four predominant high m_{lidar} regions from the low-altitude analysis, only Midtown Manhattan remains. A pattern emerges at this altitude that suggests the potential for GPS to complement Lidar, and vice-versa. Areas of low m_{gps} due to the urban canyon coexist with high m_{lidar} areas, and low m_{lidar} due to the absence of nearby obstacles results in high m_{gps} areas without satellite obstruction. This effect becomes

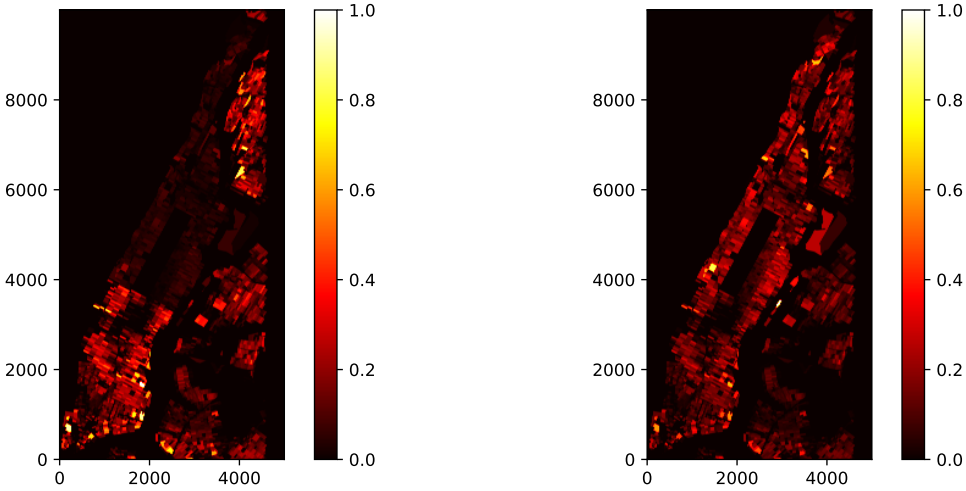


(a) Lidar 2m res map at 20m. (b) Lidar 2m res map at 60m. (c) Lidar 2m res map at 122m.

Figure 3.5: Lidar metric maps for low, medium, and high-altitude urban flight.

more apparent at high-altitude flight and above.

Day and night population metric maps, shown in Fig. 3.6, are independent of flight altitude. The following daytime population scaling factors were used: $\Gamma_{comm} = 3.0$ and $\Gamma_{resi} = 0.5$. These values are biased toward a net population influx into Manhattan for the workday as show in Table 3.6. The population map results validate the expected residence-to-work and work-to-residence commuting patterns and constraints discussed in Sec. 3.4.4.



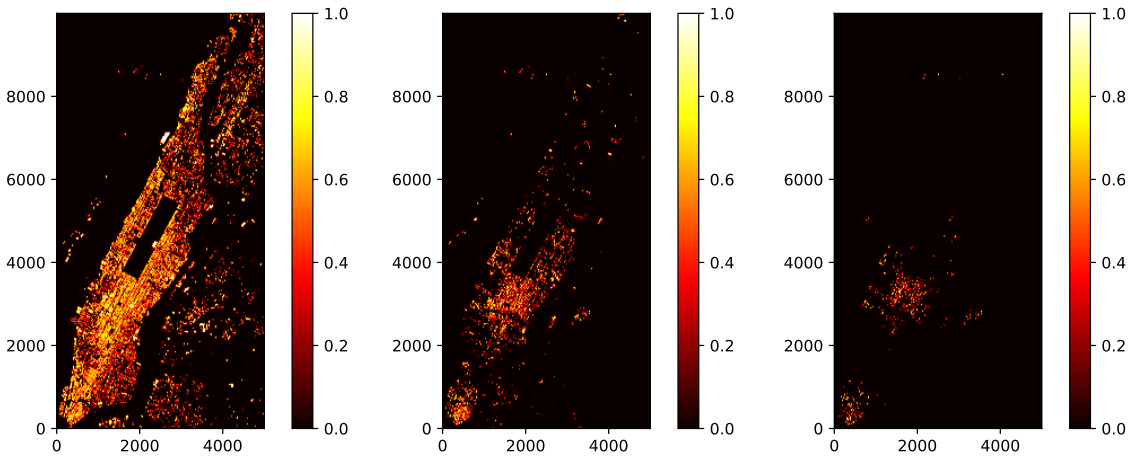
(a) Daytime population (2m res) (b) Population at night (2m res)

Figure 3.6: Population metric maps over Manhattan for day and night hours.

Table 3.6: Work weekday and nighttime population estimates in millions.

	Residential	Commericial
Daytime	0.48	3.96
Nighttime	0.97	1.32

Proximity risk maps identify obstacle-free map grid points with decaying risk value over a distance d_{thresh} around buildings, the risk is one at the building, linearly decreasing to 0 at d_{thresh}. High proximity risk areas are mostly in the Manhattan borough, as shown in Fig. 3.7. For low altitude-flight, except for the Hudson River, New Jersey, and Central Park, a building can be found within 10m in most grids. Large portions of the Bronx, Queens, Brooklyn, and Uptown Manhattan become risk-free zones at medium-altitude flight. Only Downtown and Midtown Manhattan remain at high-altitude flight due to the congestion of tall buildings, as discussed earlier.



(a) Risk at 2m res at 20m. (b) Risk at 2m res at 60m. (c) Risk at 2m res at 122m.

Figure 3.7: Proximity risk metric maps for low-altitude and medium-altitude flight.

3.6 Conclusion

This chapter has defined a set of map-based, path-based, and software-based metrics for sUAS urban flight planning. Map-based metrics were investigated in detail with metric maps generated over Manhattan at three different resolutions for four sUAS AGL flight altitudes. Results demonstrate the complementary nature of GPS and Lidar accuracy in an urban canyon as a function of altitude. By generating these metric maps a priori, an sUAS can predict risk and sensor value before a flight, i.e., GPS will provide valid position data if $m_{gps} > m_{lidar}$; Lidar will offer better data otherwise.

Population metric maps support residence-to-work and work-to-residence commuting patterns using as simplified as work-week daytime and nighttime models. In the future, this model should be extended to weekends with a time-based population function offering more resolution over 24-hour population patterns. When deep in the urban canyon, proximity-based risk is high, but it quickly decreases at higher altitudes due to fewer obstacles. For path planning, if risk is the primary cost, data indicate that flying to a higher altitude is preferable. 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.

CHAPTER IV

Algorithm Selection for UAS Motion Planning

This chapter formulates and proposes two machine learning-based solutions to the algorithm selection problem (ASP) for small UAS motion planning with a focus on low-altitude flight above a complex urban landscape. For this investigation, we assume atmospheric conditions have no negative impact on flight, geospatial data is available and accurate, no onboard failures occur, no other aircraft are in the area, and all motion planner implementations execute consistently with algorithm specifications.

Path planners generate a solution $\boldsymbol{\zeta}$ over a sequence of configurations $\zeta_i, \ldots, \zeta_n$ (positions and orientations) from initial to goal configuration [88]. Surveys [89, 90] examine techniques and analyze their relative performance. *Geometric-based* planners rapidly construct paths using points, lines, and arcs [55, 91]. Dubins [54] and Reeds [92] paths account for systems with turning constraints. Assumptions such as an obstacle-free environment may be required. *Graph-based* planners use roadmaps to define obstacle-free routes with discrete search over route segment sequences. A* [93] and its variants (Dijkstra [94], LPA* [95], ARA* [96], D* Lite [97], Field D* [98], Theta* [99]) are commonly applied. Graph-based planners are optimal but computationally-intensive. *Sampling-based* planners such as probabilistic roadmaps [100] and rapidly exploring random trees (RRTs) [101, 102] offer improved convergence

with asymptotically optimal variants (PRM*, RRT* [56]) and improved convergence rates using FMT* [103] and BIT* [3]. Sampling algorithms are probabilistically complete but may not find solutions through narrow passages. *Control-centric* solvers assure dynamics constraint satisfaction [104]. Optimal control [105] minimizes time, energy, and obstacle avoidance costs over smooth paths but with high computational complexity. Model predictive control [106, 107, 108] limits analysis to a finite horizon and can use lookup tables. Both may encounter local minima.

The algorithm selection problem (ASP) was first defined in [32]. Automated algorithm selection has been studied for propositional satisfiability [109], combinatorial search [1], and continuous optimization [110]. To our knowledge, this thesis is first to address autonomous ASP for UAS motion planning.

4.1 Motion Planning ASP Definition

Let $\mathcal{A}$ be a portfolio of motion planners available to solve problem instances $\mathcal{P}$. For each $p \in \mathcal{P}$, there exists a most suitable algorithm $a \in \mathcal{A}$. An algorithm selection ranking function $\mathcal{S} : \mathcal{P} \mapsto \mathcal{A}^*$ specifies a planner preference ranking $\mathcal{A}^*$ for each p without the need to run all algorithms in $\mathcal{A}$ to find the best solution. $\mathcal{S}$ orders all planners expected to meet specified time and memory constraints in $\mathcal{A}^*$ to determine a^*, the motion planner expected to perform best for problem instance p. $p \in \mathcal{P}$ is defined over the septuple $< \mathcal{B}, \mathcal{Q}_S, \mathcal{Q}_G, \mathcal{D}, \mathcal{H}, \mathcal{I}, \mathcal{W} >$, where $\mathcal{B}$ is a designated operating bounding box, $\mathcal{Q}_S$ & $\mathcal{Q}_G$ are start and goal configurations, respectively, $\mathcal{D}$ are current date and time, $\mathcal{H}$ is an environment model or map, $\mathcal{I}$ is aircraft information (type and properties), and $\mathcal{W}$ is a set of cost metric weightings. $\mathcal{S}$ relies on the three classes of cost and constraint metrics, path-based m_p, map-based m_m, and software execution-based m_s, defined in the previous chapter.

Algorithm portfolio $\mathcal{A}$ considered in this work considers example geometric, graph-based, and sampling-based motion planners. Control-based planning options are be-

yond the scope of this dissertation. Geometric flight planning is a staple of aerial motion planning. Dubins curves [54] for fixed-wing aircraft and point-to-point flight plans for multicopters are the candidate geometric planner options. A* [55] with an admissible heuristic is the candidate graph search option. BIT* [3] inspired by RRT* [56] is the candidate sampling-based planning option. All these planners are well-established focusing innovations of this work on ASP definition and solution.

Each motion or flight planning problem instance $p \in \mathcal{P}$ is defined by:

- $\mathcal{B}$: Bounding box defining area of interest as a function of $\mathcal{Q}_S$ and $\mathcal{Q}_G$. [tuple]
- $\mathcal{Q}_S$, $\mathcal{Q}_G$: Start and goal state planning configurations. [tuple, tuple]
- $\mathcal{D}$: Date and time to estimate GPS satellite positioning and population density.
- $\boldsymbol{\mathcal{H}}$: Generated metric maps over $\mathcal{B}$, discussed in Chap. III. [2d arrays]
- $\boldsymbol{\mathcal{I}}$: Aircraft information, e.g., mass, turning radius, max climb angle. [tuple]
- $\boldsymbol{\mathcal{W}}$: Cost weighting vector used for planning. [tuple]

Specifics of the three ASP candidate motion planner types (geometric PTP, A* graph search, BIT* sampling-based) are detailed in the next section.

4.2 Planning Algorithms

4.2.1 Point-to-Point: PTP

The simplest path a multicopter can take is a direct path to its destination, i.e. point-to-point (PTP). $\boldsymbol{\Lambda}$ defines all relevant parameters for multicopter PTP flight:

- *Parameters*: $\boldsymbol{\Lambda} = (\mathcal{L}, \mathcal{Q}_S, \mathcal{Q}_G, \boldsymbol{\mathcal{H}}_{\delta_r}, \mathcal{W}, \delta_z, \delta_r)$

for a flight at constant above-ground altitude δ_z with map data resolution δ_r. Given PTP's geometric and no-obstacle assumptions, there exists no search space for this algorithm.

As detailed in Chap. III, the operating environment is described by a collection of metric maps, $\boldsymbol{\mathcal{H}}$. Each map is a rasterized metric quantification generated for a given height and resolution pair, (δ_z, δ_r), within the bounds of map search space $\mathcal{L}$.

Using the start and goal positions, or states, the path's lth waypoint map indices $(\mathcal{Q}_{k,idx}, \mathcal{Q}_{k,idy})$ with an origin $(\mathcal{L}_{x,min}, \mathcal{L}_{y,min})$ are calculated as:

$$\mathcal{Q}_{l,idx} = \left\lfloor \frac{\mathcal{Q}'_{k,x} - \mathcal{L}_{x,min}}{\delta_r} \right\rfloor \quad \mathcal{Q}'_{l,x}(\alpha_{k,x}) = \mathcal{Q}_{S,x} + \alpha_{l,x} \tag{4.1}$$

$$\mathcal{Q}_{l,idy} = \left\lfloor \frac{\mathcal{Q}'_{k,y} - \mathcal{L}_{y,min}}{\delta_r} \right\rfloor \quad \mathcal{Q}'_{l,y}(\alpha_{k,y}) = \mathcal{Q}_{S,y} + \alpha_{l,y} \tag{4.2}$$

where $\alpha_{k,x}$ and $\alpha_{k,y}$ are component-wise steps from $\mathcal{Q}_S$ to $\mathcal{Q}_G$ for $l = 0, 1, \ldots \lceil \frac{\lambda}{\delta_r} \rceil$:

$$\alpha_{l,x} = \begin{cases} \lambda \cos(\theta) & \text{if } l = \lceil \frac{\lambda}{\delta_r} \rceil \\ l\delta_r \cos(\theta) & \text{otherwise} \end{cases} \qquad \alpha_{k,x} = \begin{cases} \lambda \sin(\theta) & \text{if } l = \lceil \frac{\lambda}{\delta_r} \rceil \\ l\delta_r \sin(\theta) & \text{otherwise} \end{cases} \tag{4.3}$$

where $\theta = \text{atan2}(\mathcal{Q}_{G,y} - \mathcal{Q}_{S,y}, \mathcal{Q}_{G,x} - \mathcal{Q}_{S,x})$ and $\lambda = \sqrt{(\mathcal{Q}_{G,y} - \mathcal{Q}_{S,y})^2 + (\mathcal{Q}_{G,y} - \mathcal{Q}_{S,y})^2}$.

Since altitude is on a number line, the z-component $\mathcal{Q}_{l,z}$ and index $\mathcal{Q}_{l,idz}$ for the l^{th} state are:

$$\mathcal{Q}_{l,z} = \delta_z \mathcal{Q}_{l,idz} = \lfloor \delta_z \rfloor \tag{4.4}$$

defined with respect to unit resolution.

To test path validity, the path is masked onto the obstacle map $\mathcal{H}_{obs,\delta_r}$. If any masked index has a non-zero value, i.e., $\mathcal{H}_{obs,\delta_r}(\mathcal{Q}_{l,idx}, \mathcal{Q}_{l,idy}, \mathcal{Q}_{l,idz}) > 0$, the path ζ is

invalid; otherwise its cost is calculated. Total path cost is defined by:

$$f(\zeta) = \sum_{(\mathcal{Q}_{i-1}, \mathcal{Q}_i) \in \zeta} c(\mathcal{Q}_{i-1}, \mathcal{Q}_i) \tag{4.5}$$

where $c(\cdot)$ is the transition cost between two states, assuming a transition is possible, and the path ζ is a sequence of states $(\mathcal{Q}_S, \mathcal{Q}_1, \ldots, \mathcal{Q}_G)$.

When using grid-based maps, the cost of moving between grids is described by a collection of k cost maps $\boldsymbol{\mathcal{H}}_{\delta_r}$ with the same dimensions as the total search space $\mathcal{L}$ at an altitude δ_z with resolution δ_r. Given a state's indices, e.g, (idx, idy, idz), one can calculate the k map-based costs for entering said state. Supplied with a weighting scheme $\mathcal{W}$, the map-based costs can be combined using a weighted sum model.

The transition costs between two neighboring states $\mathcal{Q}_a$ to $\mathcal{Q}_b$ is:

$$c(\mathcal{Q}_a, \mathcal{Q}_b) = w_0 d_{euc}(\mathcal{Q}_a, \mathcal{Q}_b) + \sum_{j=1}^{k} w_j \mathcal{H}_j(\mathcal{Q}_{b,idx}, \mathcal{Q}_{b,idy}, \mathcal{Q}_{b,idz}) \tag{4.6}$$

where $d(\cdot)$ is the Eucledian distance between states, w_0 is the distance weight, and w_k for $j = 1, \ldots, k$ are the map cost weights such that $\sum_{j=0}^{k} w_j = 1$ and $w_j \in \mathcal{W} \mid \forall j$.

4.2.2 Graph-based Planning: A*

A* [93] is a discrete graph-based informed search algorithm popular for its completeness, optimality, and spatial efficiency. A* searches a graph $\mathcal{G}$, composed of nodes N and edges E, to find a sequence of edge transitions that optimally navigates $\mathcal{G}$ from a start node $\mathcal{Q}_S$ to a goal node $\mathcal{Q}_S$. In motion planning, this sequence of edge transitions is equivalent to the desired path ζ.

The A* motion planning problem is defined by:

- *Parameters*: $\boldsymbol{\Lambda} = (\mathcal{L}, \mathcal{Q}_S, \boldsymbol{\mathcal{H}}_{\delta_r}, \mathcal{Q}_G, \mathcal{W}, \delta_z, \delta_r, \delta_c)$
- *Search Graph*: $\mathcal{G} = (V(\cdot), E(\cdot))$

- *Cost Function*: $f(\cdot) = g(\cdot) + h(\cdot)$

where δ_c is the connection logic used to create the search graph $\mathcal{G}$, (V, E) are the nodes and edges forming $\mathcal{G}$, $g(\cdot)$ is the cost function from the start node to the current search node, and $h(\cdot)$ is a heuristic function estimating cost from the current search node to the goal node.

Node vertices V are defined by discretizing $\mathcal{L}$ with resolution δ_r. In an obstacle-free environment a total of $\frac{\mathcal{L}_{dx}\mathcal{L}_{dy}}{\delta_r^2}$ are realizable, assuming the remainders of $\mathcal{L}_{dx}/\delta_r = \mathcal{L}_{dy}/\delta_r = 0$. Under these contraints, the total potential configuration space $\mathcal{C}_{tot}$ is:

$$\mathcal{C}_{tot} = \left\{ \mathcal{Q}_l \mid \forall i, j \wedge \mathcal{Q}_{l,x} = i\frac{\mathcal{L}_{dx}}{\delta_r} \wedge \mathcal{Q}_{l,y} = j\frac{\mathcal{L}_{dy}}{\delta_r} \right\} \tag{4.7}$$

where $i = 1, 2, \ldots, \mathcal{L}_{dx}/\delta_r$ and $j = 1, 2, \ldots, \mathcal{L}_{dy}/\delta_r$ such that $l = (i-1)\mathcal{L}_{dy}/\delta_r + j$.

However, in the presence of obstacles some nodes may be invalid. Nodes with obstacle conflicts given by $\mathcal{C}_{obs} \subseteq \mathcal{C}_{tot}$ is defined as:

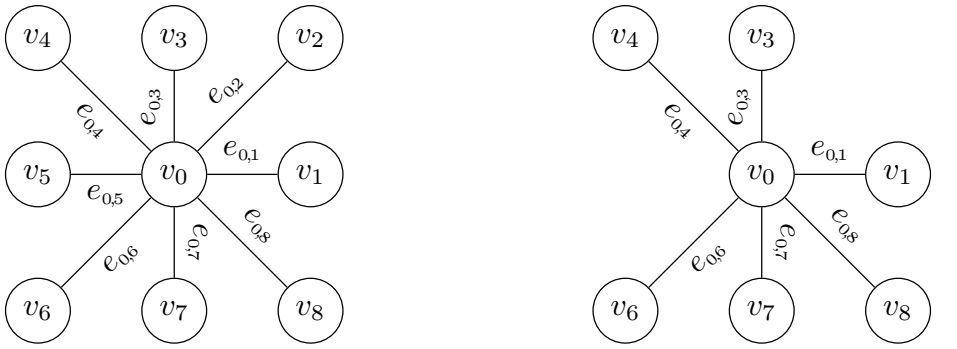
$$\mathcal{C}_{obs} = \{\mathcal{Q} \mid \mathcal{Q} \in \mathcal{C}_{tot} \wedge \mathcal{H}_{obs,\delta_r}(\mathcal{Q}_{idx}, \mathcal{Q}_{idy}, \mathcal{Q}_{idz}) = 1\} \tag{4.8}$$

Having calculated $\mathcal{C}_{tot}$ and $\mathcal{C}_{obs}$, the graph nodes can be defined as:

$$V = \mathcal{C}_{tot} \setminus \mathcal{C}_{obs} \equiv \mathcal{C}_{free} \tag{4.9}$$

where $\mathcal{C}_{free}$ is the configuration space of obstacle-free nodes.

Graph edges can be created for all neighboring nodes as defined by the connection logic δ_c. For an *8-connected* logic, any node v_0 has potential neighbors $v_1, v_2, \ldots, v_8$ as shown in Fig. 4.1a, with non-diagonal (odd) and diagonal (even) edges. Due to obstacles, not all neighbors might be reachable, as shown in Fig. 4.1b where we assume $v_2, v_5 \in \mathcal{C}_{obs}$ for demonstration purposes.



(a) Fully connected graph. (b) Partially connected graph.

Figure 4.1: Graph nodes, edges, and costs with 8-connected logic.

Accounting for obstacles, all feasible graph edges can be computed as follows:

$$E = \left\{ e_{m,n} = (\mathcal{Q}_m, \mathcal{Q}_n) \in \binom{V}{2} \right\} \tag{4.10}$$

where i and j serve as node identifiers or IDs.

Having defined the graph $\mathcal{G} = (V, E)$, the start and goal nodes need to be appended and labeled. In this investigation, we do the latter by finding the nodes in $\mathcal{G}$ closest to $\mathcal{Q}_S$ and $\mathcal{Q}_G$ and labels are assigned appropriately. An optimal path can now be constructed using A* search on $\mathcal{G}$.

To optimize path construction, A* uses the total cost $f(\mathcal{Q}_n) = g(\mathcal{Q}_n) + h(\mathcal{Q}_n)$ where $g(\mathcal{Q}_n)$ is the cumulative cost *cost-sor-far* from $\mathcal{Q}_S$ to $\mathcal{Q}_n$, and $h(\mathcal{Q}_n)$ is a heuristic function estimating the *cost-to-go* from $\mathcal{Q}_n$ to $\mathcal{Q}_G$. The *cost-so-far* is a cummulative cost analysis of node transtions as up to $\mathcal{Q}_n$ such that:

$$g(\mathcal{Q}_n) = g(\mathcal{Q}_m) + c(\mathcal{Q}_m, \mathcal{Q}_n) \tag{4.11}$$

where $\mathcal{Q}_m$ the parent node of $\mathcal{Q}_n$ Eq. 4.6 is used to calculate the function $c(\cdot)$.

Built on the underlying optimalty of Dijkstra's algorithm [94], the A* heuristic function $h(\cdot)$ maintains optimality and improves search efficiency so long as:

- $h(\cdot)$ is admissible, i.e., it never overestimates the true *cost-to-go*.

- $h(\cdot)$ is consistent, i.e., for any successor configuration n, $h(m) \leq c(m,n) + h(n)$, where $c(\cdot)$ is the true cost to travel from m to n.

Under these conditions, we propose the following heuristic:

$$h_{plus}(\mathcal{Q}_i) = w_0 \hat{d}(\mathcal{Q}_i, \mathcal{Q}_G) + \sum_{j=1}^{k} w_j \hat{s}_j(\mathcal{Q}_i) \tag{4.12}$$

where $\hat{d}(\cdot)$ approximates the remaining distance to the goal and $\hat{s}_j(\mathcal{Q}_i)$ conservatively estimates the cummulative k map-based costs described in Sec. III for the final path.

The distance function $\hat{d}(\cdot)$ is chosen to be admissable by default. For an 8-connected uniform grid, octile distance gives the minimum distance between any node pair. Octile distance d_{oct} extends Manhattan distance by allowing for diagonal transitions. The octile distance between two nodes, m, n can be computed as:

$$d_{oct}(m,n) = D_{stan}(dx + dy) + (D_{diag} - 2D_{stan}) \min(dx, dy) \tag{4.13}$$

where $dx = \text{abs}(n_x - m_x)$ and $dy = \text{abs}(n_y - m_y)$ represent the maximum number of horizontal dx and vertical dy transitions required to reach node n from m.

Next, using the information encoded by each map $\mathcal{H} \in \boldsymbol{\mathcal{H}}_{\delta_r}$ we estimate the minimum map-based costs for any path heading to $\mathcal{Q}_G$. Assuming current node is $\mathcal{Q}_i$ an axis-aligned bounding box (AABB) $\mathcal{B}$ is constructed such that:

$$\mathcal{B} = \begin{pmatrix} \min(\mathcal{Q}_{i,x}, \mathcal{Q}_{G,x}) \\ \min(\mathcal{Q}_{i,y}, \mathcal{Q}_{G,y}) \\ \max(\mathcal{Q}_{i,x}, \mathcal{Q}_{G,x}) \\ \max(\mathcal{Q}_{i,y}, \mathcal{Q}_{G,y}) \end{pmatrix} = \begin{pmatrix} \mathcal{B}_{x,min} \\ \mathcal{B}_{y,min} \\ \mathcal{B}_{x,max} \\ \mathcal{B}_{y,max} \end{pmatrix} \tag{4.14}$$

with $n_r = \text{abs}(\mathcal{B}_{y,max} - \mathcal{B}_{y,min})/\delta_r$ rows and $n_c = \text{abs}(\mathcal{B}_{x,max} - \mathcal{B}_{x,min})/\delta_r$ columns.

The column and row index mappings between $\mathcal{B}$ and $\mathcal{L}$ are computed as follows:

$$\mathcal{B}_{l,idx} = \left\lfloor \frac{\mathcal{B}_{y,min} + i\delta_r - \mathcal{L}_{y,min}}{\delta_r} \right\rfloor \qquad \mathcal{B}_{l,idy} = \left\lfloor \frac{\mathcal{B}_{x,min} + j\delta_r - \mathcal{L}_{x,min}}{\delta_r} \right\rfloor \tag{4.15}$$

for $i = 0, 1, \ldots, n_c$ and $j = 0, 1, \ldots n_r$.

Using the index bounds for rows $(\mathcal{B}_{0,idy}, \mathcal{B}_{n_r,idy})$ and columns $(\mathcal{B}_{0,idx}, \mathcal{B}_{n_c,idx})$ the ith row or jth column used by the heuristic can be expressed as:

$$\mathcal{C}_{k,j} = \{\mathcal{H}_k(\mathcal{B}_{0,idy}, \mathcal{B}_{j,idx}), \mathcal{H}_k(\mathcal{B}_{1,idy}, \mathcal{B}_{j,idx}), \ldots, \mathcal{H}_k(\mathcal{B}_{n_r,idy}, \mathcal{B}_{j,idx})\} \tag{4.16}$$

$$\mathcal{R}_{k,i} = \{\mathcal{H}_k(\mathcal{B}_{i,idy}, \mathcal{B}_{0,idx}), \mathcal{H}_k(\mathcal{B}_{i,idy}, \mathcal{B}_{1,idx}), \ldots, \mathcal{H}_k(\mathcal{B}_{i,idy}, \mathcal{B}_{n_c,idx})\} \tag{4.17}$$

for the kth type of cost map $\mathcal{H}_k \in \boldsymbol{\mathcal{H}}_{\delta_r}$.

The minimum cost for the kth map-based metric is computed as follows:

$$s_k(n) = \max\left(\sum_{i=1}^{dx} \min(\mathcal{C}_{k,i}), \sum_{i=1}^{dy} \min(\mathcal{R}_{k,i})\right) \tag{4.18}$$

By construction, this portion of the heuristic is consistent, hence admissable. Since both portions of the heuristic are admissible, then the overall presented heuristic is admissible as well, guaranteeing optimality. To test the novel heuristic, two A* variants are studied in this dissertation. A$^*_{dist}$ uses a traditional Euclidean distance-to-goal heuristic h_{dist} while A$^*_{plus}$ applies the novel h_{plus} defined in Eq. 4.12.

4.2.3 Sampling-based Planning: BIT*

Batch Informed Trees (BIT*) [3] is a sampling-based search algorithm that improves scalability relative to classical graph-based techniques. Extending on previous work [57], BIT* utilizes an iterative search graph $\mathcal{G}$ informed by previous solutions. When a solution is found, BIT* reduces its search space $\mathcal{C}_{free}$, prunes and reuses its search graph, generates a new set of samples in the new $\mathcal{C}_{free}$, and restarts its search.

BIT* terminates when a cost threshold has been met or all δ_b batches are complete.

For this investigation, the BIT* motion planning problem is defined by:

- *Parameters*: $\mathbf{\Lambda} = (\mathcal{L}, \mathcal{Q}_S, \mathcal{Q}_G, \boldsymbol{\mathcal{H}}_{\delta_r}, \mathcal{W}, \delta_z, \delta_r, \delta_b, \delta_s)$
- *Search Tree*: $\mathcal{T}_i = \text{BIT}^*\ (\mathcal{T}_{i-1}, \mathcal{H}_{obs,\delta_r}, \delta_s)$ for $i = 1, 2, \ldots, \delta_b$
- *Total Cost Function*: $f(\cdot) = g(\cdot) + h(\cdot)$

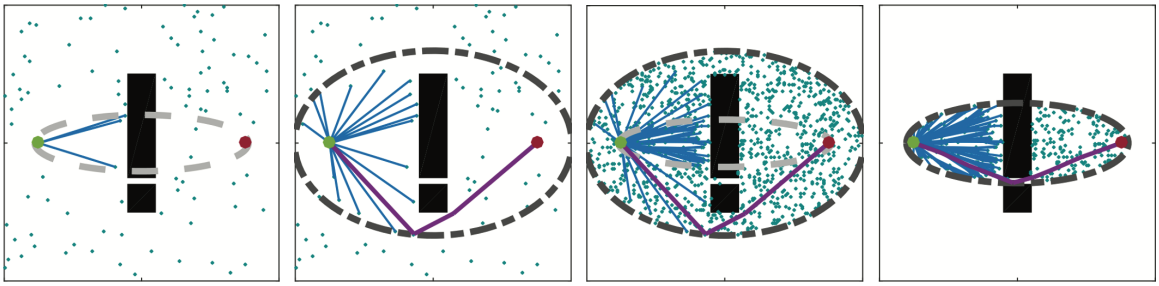
where BIT*$(\cdot)$ returns a graph, and path if found, updated with δ_s samples per batch.

Similar to A*, BIT* uses a *cost-so-far* function $g(\cdot)$ and *cost-to-go* heuristic $h(\cdot)$ to search a series of increasingly dense implicit rapidly-exploring random graphs (RRGs) efficiently as illustrated in Fig. 4.2. When initializing the ith batch, the search for a solution expands outward from the minimum cost solution, adding feasible connections from $\mathcal{C}_{free,i}$ to a growing tree T_i with nodes and edges (V_i, E_i). If a solution is found, the batch ends and the search space $\mathcal{C}_{free,i+1}$ is redefined so new samples can only improve the current solution. The previous tree is pruned of any nodes and edges outside of $\mathcal{C}_{free,i+1}$ such that:

$$V_{i+1} = V_i \cap \mathcal{C}_{free,i+1} \qquad E_{i+1} = \{e_{m,n} \mid e_{m,n} \in E_i \wedge \mathcal{Q}_m, \mathcal{Q}_n \in V_{i+1}\} \tag{4.19}$$

A new set of δ_s nodes are sampled in $\mathcal{C}_{free,i+1}$ and the search restarts for the next batch. BIT* terminates when all δ_b batches are complete, or the latest solution meets some cost-ending criteria, e.g., a percent change or total cost threshold.

During the first batch, T_1 is initiated such that $V = \{\mathcal{Q}_S\}$ and $E = \emptyset$. Nodes are added to the closest node in the current tree if a collision-free edge is feasible and they improve the best solution so far $\hat{\zeta}$. The costs of of adding a new node $\mathcal{Q}_n$ with an edge $e_{m,n}$ are computed using Eq. 4.11 for $g(\mathcal{Q}_n)$ and Eq. 4.6 for $c(\mathcal{Q}_m, \mathcal{Q}_n)$, respectively. Similar to the A* variants, BIT*$_{dist}$ uses h_{dist} as its heuristic while BIT*$_{plus}$ applies the novel h_{plus} defined in Eq. 4.12.



(a) For each batch, the search expands out from the minimum solution.

(b) When a solution is found, the batch finishes and a new search space is defined.

(c) A new batch of samples is added to a newly reduced search space and restarts.

(d) The process repeats to find a better solution every batch.

Figure 4.2: BIT* batch process as adapted from [3].

4.3 Monte Carlo Planning Results Summary

4.3.1 Simulation Procedure

All map generation and planning simulations were performed using the Google Cloud: Compute Engine (CE). The Google Cloud CE hosts general-purpose, computation optimized, and memory-intensive high-end virtual machines depending on the user's demands. In this study, all maps and Monte Carlo planning simulations were generated using ten *n1-standard-16* virtual machines (VMs). Each VM consists of 16 vCPUs with an underlying architecture of Intel(R) Xeon(R) CPU @ 2.30GHz with 60GB memory. A rate of 0.031611 USD per vCPU hour is charged per VM.

Two geospatial datasets were used for all simulations: (1) OSM and (2) TIGER. OSM data was downloaded from PlanetOSM[†] as 50+ GB PBF file. TIGER 2010 US Census data was downloaded directly from the US Census Bureau as a 180+ MB shapefile. The Geospatial Data Abstraction Library (GDAL) was used to uncompress and extract all Manhattan-specific data within $\mathcal{L}$. Processing all raw data for metric map generation required 460+ GB for the 200km^2 area. The workload was distributed uniformly across all VMs, requiring a week's worth of runtime.

[†]https://planet.openstreetmap.org/

Start $\mathcal{Q}_S$ and goal $\mathcal{Q}_G$ configurations were uniformly sampled across $\mathcal{L}$ to capture all relevant subdomains, e.g., flying over water, suburban, and highrise areas. 5000 configuration pairs were sampled for each altitude and resolution combination for a total of 60000 pairs. Fig. 4.3 showcases the triangularly distributed ranges of the sampled configurations, with a peak range of 7500m due to the dimensions of $\mathcal{L}$. Consistent with sUAS ranges, pair ranges were capped between 1km and 20km.

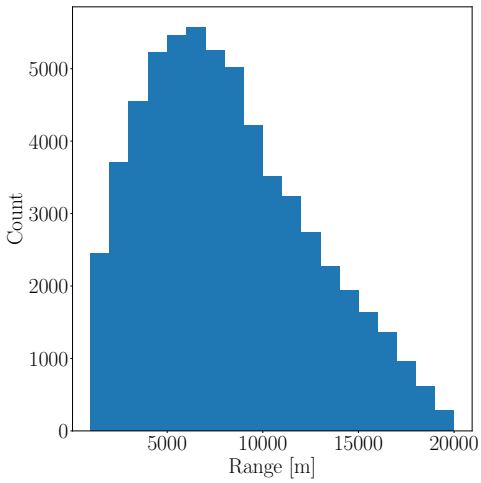


Figure 4.3: Range distribution of sampled start and goal configurations.

Weighting vectors $\mathcal{W}$ were semi-randomly generated for all problem instances. First, a weight of 0.5 was assigned to all w_{dist} for all problem instances. Second, the remaining half was distributed among $w_{gps}, w_{lidar}, w_{pop}, w_{risk}$ using one of three distinct techniques: (1) uniform sampling [50%], (2) random selection [25%], (3) equal weighting [25%], with frequency in brackets. These three techniques were chosen to best showcase all possible weightings of interest. 5000 weighting vectors were sampled for each altitude and resolution combination, for a total of 60000 cases.

Each motion planning algorithm in $\mathcal{A}$ was implemented as discussed in Sec. 4.2 in Cython, Python's optimized statically compiled variant. Cython takes advantage of Python's high-level, easily readable syntax while providing speeds comparable to C/C++ on execution. All problem instances in $\mathcal{P}$ were equally distributed among all VMs and ran against each planner. Planner execution results were transferred and

processed locally, as discussed in the following section.

4.3.2 Cost and Execution Time Results

Three benchmarks are considered to analyze planner performance: (1) Expected cost $cost(a, p_i)$, (2) Execution time $time(a, p_i)$, and (3) Success rate $sr(a, \mathcal{P})$. Expected cost $cost(a, p_i)$ is calculated for path $\zeta(a, p_i)$ returned by planner a on problem instance p_i using Eq. 4.5 to compute total path cost $f(\zeta)$. Planner execution time is recorded for all cases as $t(a, p_i)$. Success rate $sr(a, \mathcal{P})$ is the likelihood of a particular planner finding a feasible path at or before the planning deadline. All successful planner executions were compiled and statistically analyzed for their expected cost and execution time benchmarks as defined below:

$$cost(a, p_i) = \begin{cases} f(\zeta) & \text{if } \zeta \neq \emptyset \\ c_{max} & \text{otherwise} \end{cases} \tag{4.20}$$

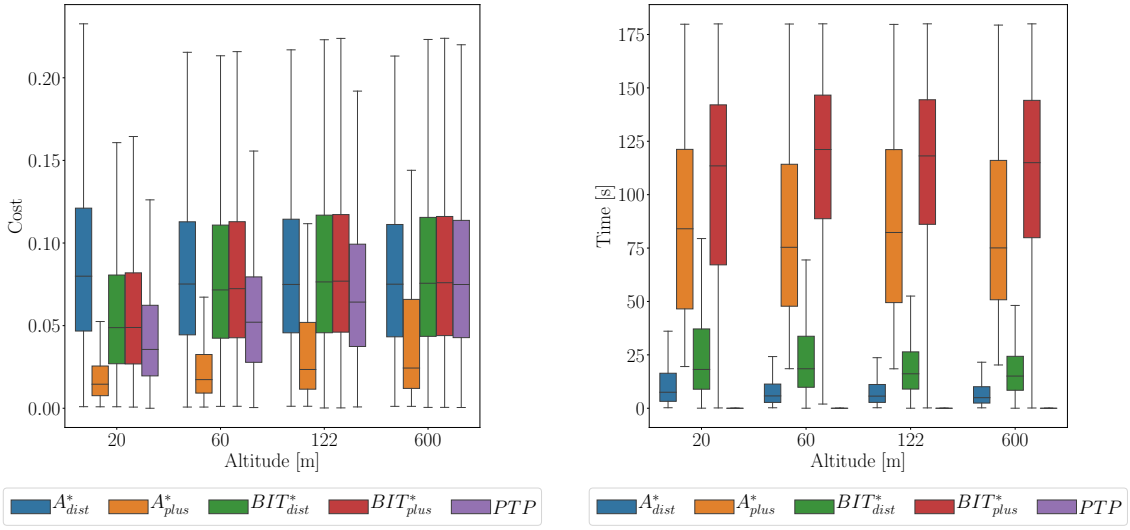
$$time(a, p_i) = \begin{cases} t(a, p_i) & \text{if } \zeta \neq \emptyset \\ t_{max} & \text{otherwise} \end{cases} \tag{4.21}$$

where c_{max} and t_{max} are the maximum recorded cost and times over all (a, p_i) pairs.

Successful planner executions are tallied by data resolution in this section. Boxplots were created for each planner with minima/maxima within $[Q1 - 1.5 \cdot \text{IQR}, Q3 + 1.5 \cdot \text{IQR}]$ to eliminate outliers, where $Q1$ and $Q3$ are the first and third data quartiles with an interquartile range (IQR) distance measurement between $Q1$ and $Q3$. Costs have been normalized with respect to the largest cost measured for any planner at that same map resolution. Execution times ranged from a few milliseconds to a predefined cutoff deadline of 180 seconds (3 minutes) for all planners.

Cost and time benchmarks are summarized in Fig. 4.4 as functions of altitude

for 2m map resolution. A^*_{plus} yields the lowest planning costs with median cost and range increasing at higher altitudes. A similar pattern is observed for PTP since longer feasible paths exist above buildings. The remaining planners do not exhibit noteworthy altitude-related differences. In contrast, BIT^*_{dist}, A^*_{dist}, and PTP had the fastest execution times. As expected, PTP was the quickest followed by A^*_{dist} (2 sec median) and BIT^*_{dist} (20 sec median) using geometry and distance-only cost functions. A^*_{plus} (90 sec median) was 33 percent faster than BIT^*_{plus} (120 sec median) when a feasible path was found.



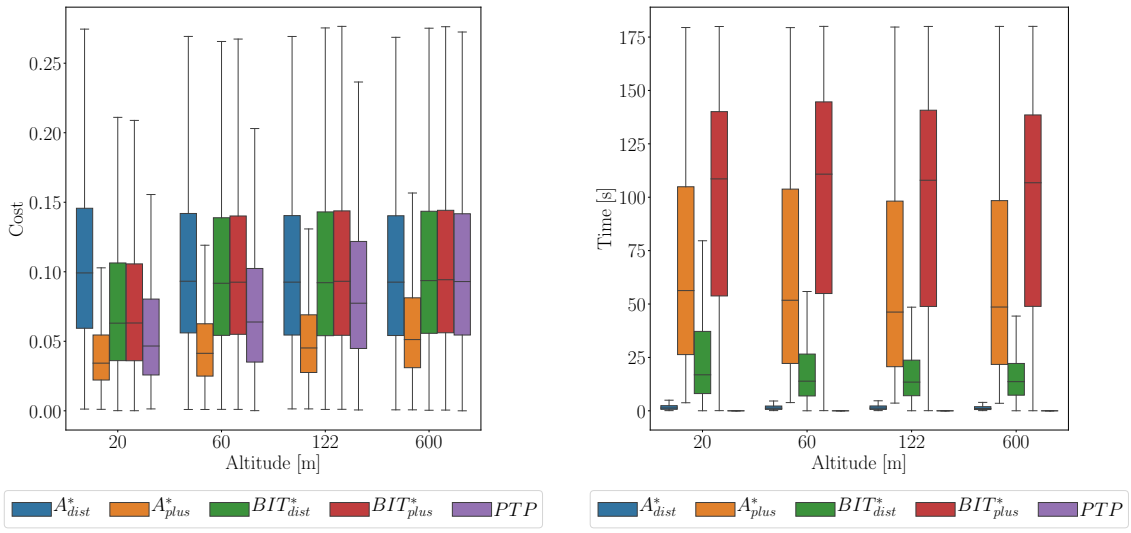
(a) Planner costs.

(b) Planner execution times.

Figure 4.4: Cost and time analysis for all planners with 2m resolution data.

Successful paths for 5m medium resolution map data in Fig. 4.5 showed similar benchmark patterns to their 2m counterparts. A^*_{plus} produced the lowest median cost paths. One noticeable distinction was the overall increase in median path costs for all planners relative to 2m resolution results. The lower 5m resolution reduces the likelihood of finding the lowest-cost path since map-based costs are now averaged over a larger area per grid. Median time benchmarks did not significantly changes except for speed-up in A* variants since larger cells reduce search tree depth.

The medium resolution planner benchmarks are magnified for 10m resolution data



(a) Planner costs.

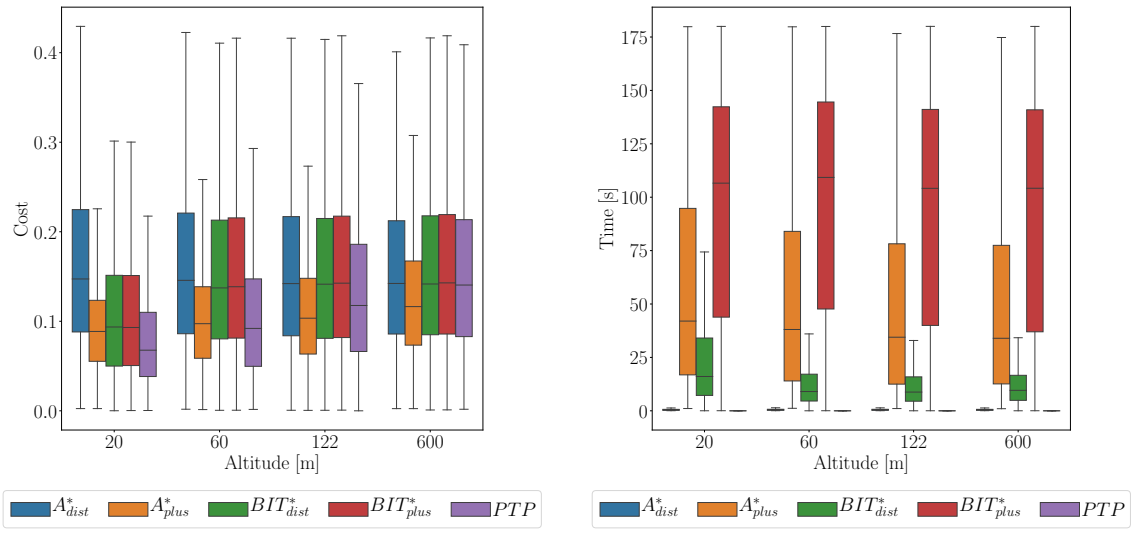
(b) Planner execution times.

Figure 4.5: Cost and time analysis for all planners with 5m resolution data.

as shown in Fig. 4.6. Median path costs for all planners have increased, but differences in median costs between planners have decreased. While still existent, the A^*_{plus} minimum cost advantage is no longer as significant. Time benchmarks have once again decreased overall. A^*_{dist} is now close to being on par with PTP, and median A^*_{plus} times range between 35-40 seconds, about half of the execution time values recorded at 2m resolution. This data illustrates the tradeoff between cost and execution time with respect to data resolution. While higher resolution data minimizes path costs and lower resolution data minimizes cost, this effect is most pronounced in search-based motion planners with complex cost maps.

4.3.3 Success Likelihood Results

Success rate sr is the likelihood of a planner finding a feasible path before the planning deadline. In any trial for which the planner cannot find a path or goes overtime, the algorithm-instance pair $(a \in \mathcal{A}, p \in \mathcal{P})$ is labeled as a failure; it is labeled a success otherwise. Success rate sr of an algorithm a is defined as the ratio



(a) Planner costs.

(b) Planner ececution times.

Figure 4.6: Cost and time analysis for all planners with 10m resolution data.

of successfully found paths S_i for a subset of problem instances $\hat{\mathcal{P}} \in \mathcal{P}$:

$$sr(a, \hat{\mathcal{P}}) = \frac{\text{card}(\{S_i(a, p_i) = success \mid \forall p_i \in \hat{\mathcal{P}}\})}{\text{card}(\hat{\mathcal{P}})} \tag{4.22}$$

where $\text{card}(\cdot)$ is the set cardinality operator. The binary-valued function $S_i(a, p_i)$ is defined by:

$$S_i(a, p_i) = \begin{cases} success & \text{if } \zeta(a, p_i) \text{ exists} \\ failure & \text{otherwise} \end{cases} \tag{4.23}$$

where $\zeta(a, p_i)$ was previously defined as the solution (path plan) returned from executing planner a on problem instance p_i.

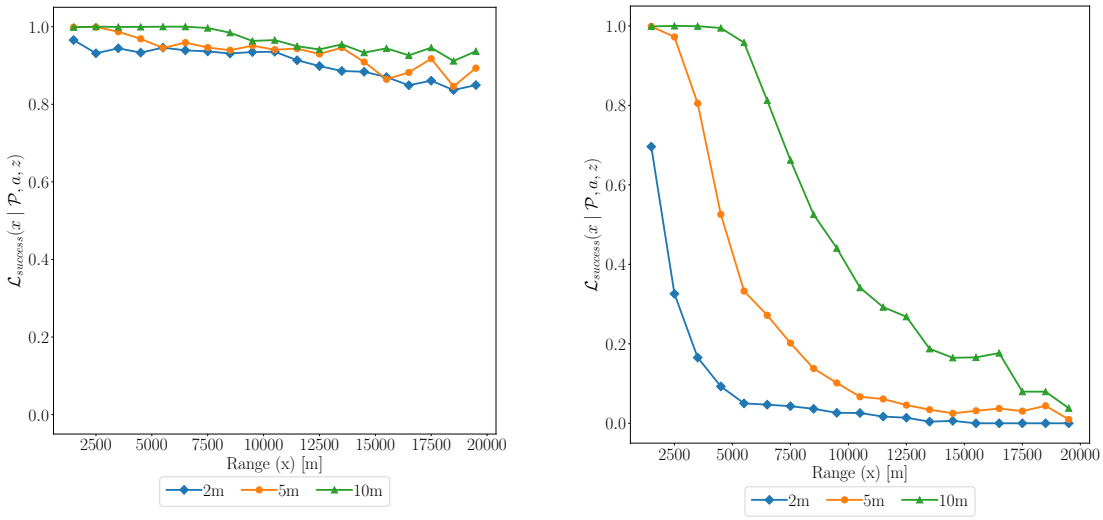
Success rates were primarily influenced by planning problem instance altitude and range $d_{euc}(\mathcal{Q}_s, \mathcal{Q}_G)$. Data resolution, as shown in Table 4.1, had little to no effect on motion planner success rate except for the A* variants that show improved success rate benchmarks as resolution decreased due to reduced search tree depth.

Focusing on range, Fig. 4.7 shows the tabular result for the A* variants. Most

Table 4.1: Planner success rates with respect to map data resolution.

Resolution	A^*_{dist}	A^*_{plus}	BIT^*_{dist}	BIT^*_{plus}	PTP
2 m	0.93	0.09	0.81	0.81	0.54
5 m	0.95	0.31	0.80	0.80	0.54
10 m	0.98	0.63	0.80	0.80	0.53

prominent for A^*_{plus} is a steep success rate decay at three ranges: 3500m (2m resolution), 4500 (5m resolution), and 8500m (10m resolution). High and medium resolution data should be used for short-range flight, while low resolution data may suffice for mid-range flight, i.e., no more than 7500m range.



(a) Success rates for A^*_{dist}.

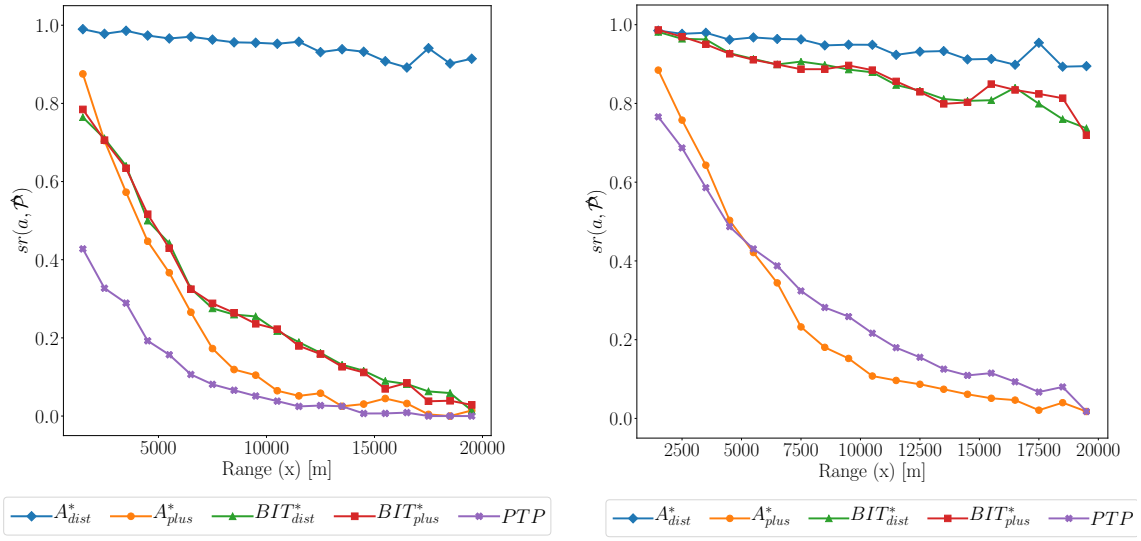
(b) Success rates for A^*_{plus}.

Figure 4.7: A* planner success rates over all planning problem instances partitioned by map resolutions of 2m, 5m, and 10m.

For a given altitude z^* and range x acting as an independent variable, Fig. 4.8 shows planner success rates sr for low and mid-altitude flight with $\hat{\mathcal{P}}(z^*, x) = \{p \in \mathcal{P} \mid z^* = 20, x\}$ and $\hat{\mathcal{P}}(z^*, x) = \{p \in \mathcal{P} \mid z^* = 60, x\}$ respectively. For low altitude (20m) flight, we observe that A^*_{dist} has the highest success rate of all planners in any scenario. In direct contrast, PTP has the worst. A^*_{plus} initially has a higher success rate than the BIT* variants and keeps a five percent margin of difference between 3000m and 6000m ranges. With greater range, all planners are less likely to find a

path due to planning time constraints and obstacles, i.e., buildings.

Mid-altitude (60m) flight showcased a similar trend for A^*_{dist} as it most efficiently navigates dense obstacle areas. However, this time around, the BIT* variants showed a drastic improvement with a worst-case success rate of about 75 percent. A^*_{plus} and PTP demonstrated similar success rate patterns, varying no more than 10 percent between each other. A^*_{plus} initially dominated PTP's success rate until about 5000m, after which PTP was more likely to find a path. With fewer obstacles at this altitude PTP more frequently returned a feasible solution.



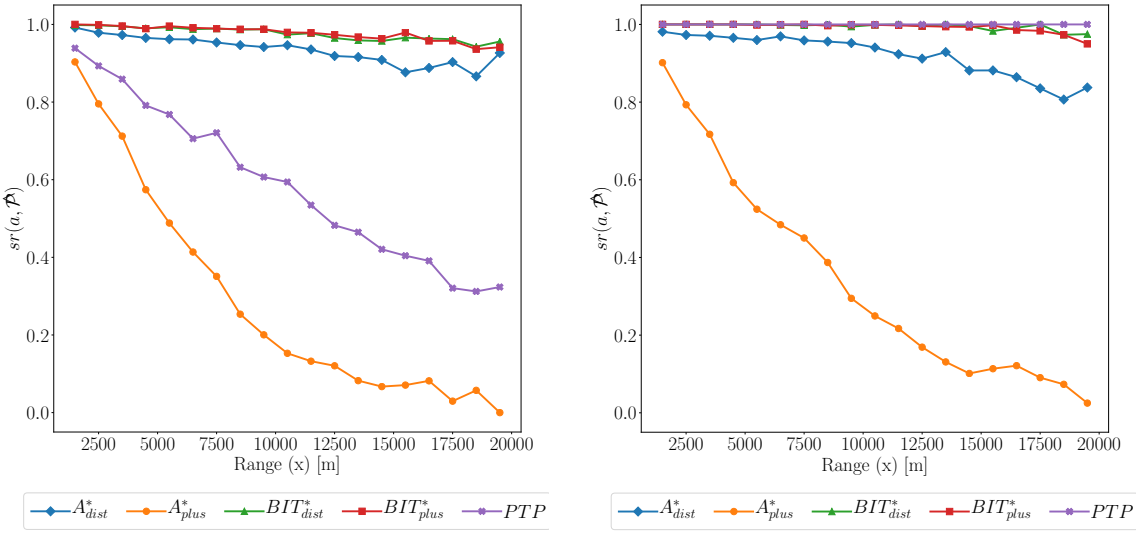
(a) Success rates at 20m altitude.

(b) Success rates at 60m altitude.

Figure 4.8: Planner success rates for low and medium altitude flight.

Fig. 4.8 shows planner success rates for high and ceiling-altitude flight where $\hat{\mathcal{P}} = \{p \in \mathcal{P} \mid z^* = 122, x\}$ and $\hat{\mathcal{P}} = \{p \in \mathcal{P} \mid z^* = 600, x\}$ respectively. For high altitude (122m) flight, BIT* outperforms A^*_{dist}. In the presense of few to no obstacles, sampling-based planners have a near perfect success rate because there are fewer collisions to manage. PTP has now surpassed A^*_{plus} for all ranges. However, it still underperforms the BIT* variants as PTP cannot circumvent the small number of obstacles still present. Above all obstacles, at 600m AGL, PTP now always finds a solution with BIT* variants a close second, occasionally failing due to path

connectivity issues at the largest ranges.



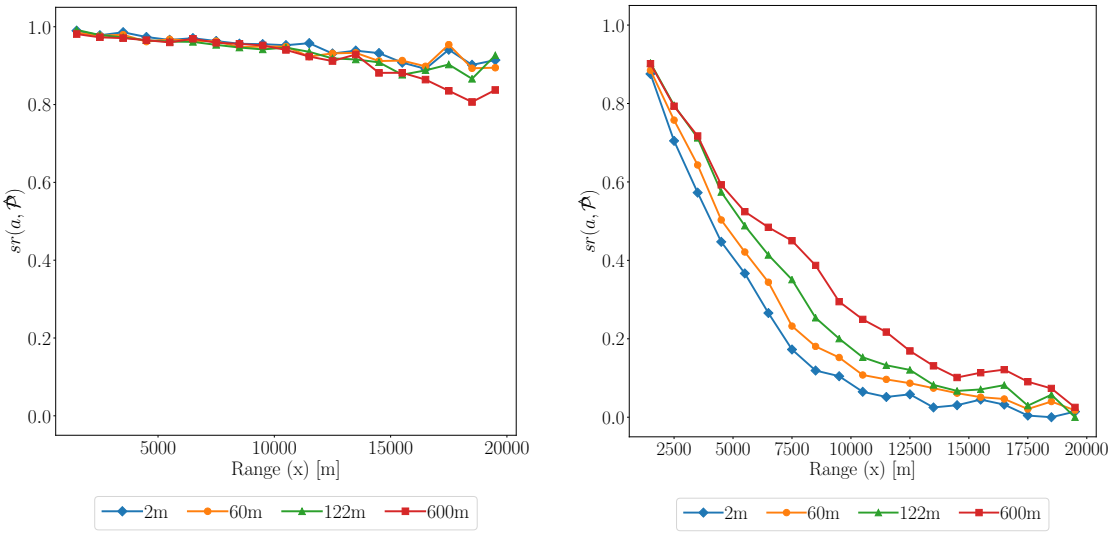
(a) Observed success rates given altitude over start to goal range at 122m altitude.

(b) Observed success rates given altitude over start to goal range at 600m altitude.

Figure 4.9: Planner success rates for high and ceiling altitude flight.

Fig. 4.10 highlights the better success rates of A^*_{dist} over A^*_{plus}. The A^*_{dist} success rate is above 80 percent for all cases, while A^*_{plus} only achieves high success rates at the lowest ranges. A^*_{dist} performs worst at 600m AGL due to a larger search space resulting in more overtime runs given high map resolution. In contrast, A^*_{plus} performs better at higher altitudes. With fewer obstacles to circumvent, its search tree breadth is reduced, arriving at a solution sooner. This is most apparent for mid-range flight (8500m $\pm$ 4000m) with negative altitude-range correlation for A^*_{plus}.

Fig. 4.11 shows success rate trends for BIT^*_{dist} and PTP at different altitudes. Note that BIT^*_{dist} and BIT^*_{plus} have near identical trends so only one plot is shown. Overall, both BIT* and PTP perform better as altitude increases due to the associated reduction in obstacles. BIT* shows a dramatic improvement for mid-altitude flight with a worst-case 80 percent success rate. This rapid success rate increase is due to BIT*'s ability to efficiently traverse a sparse obstacle set. In comparison, PTP success trends improve gradually, achieving 100 percent success at 600m when no obstacles



(a) Success rates for A^*_{dist}.

(b) Success rates for A^*_{plus}.

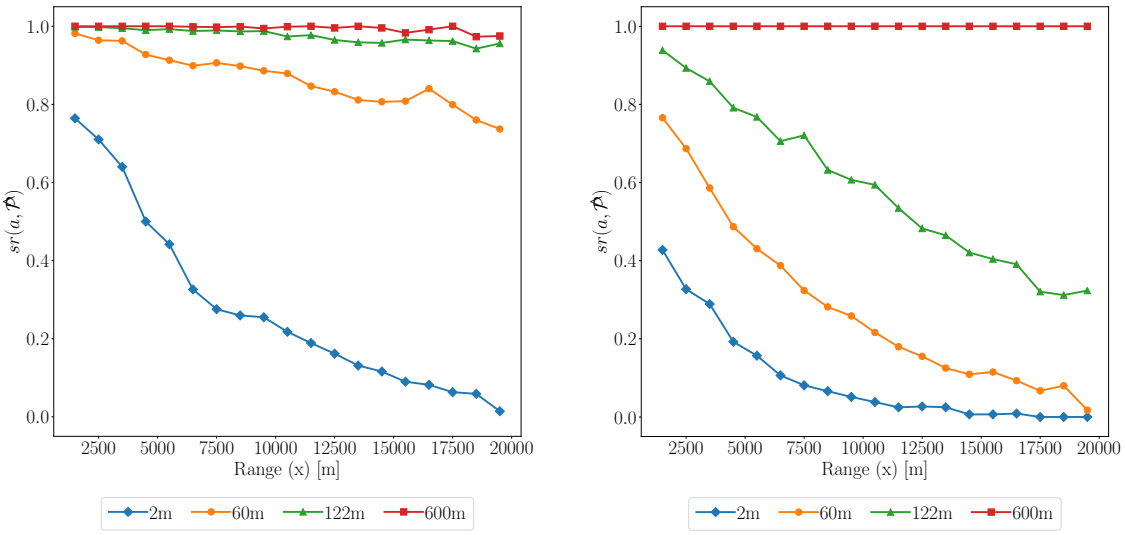
Figure 4.10: Planner success rates for A* variants for all altitudes.

are present. If distance is a vehicle's only cost metric, e.g., for high-altitude cruise flight, geometric planners such as PTP are ideal with sampling planners as reasonable alternatives.

4.4 Path Analysis

This section analyzes solution path properties from Monte Carlo simulations. Case studies are selected for each altitude $z^* \in \{20\text{m}, 60\text{m}, 122\text{m}, 600\text{m}\}$ AGL. Motion planning solutions generated within the alloted time (three minutes) are shown relative to the total unweighted cost map $\mathcal{H}_{total}$ referenced during planning. Total cost maps $\mathcal{H}_{total}(z^*)$ are defined by:

$$\mathcal{H}_{total}(z^*) = \mathcal{H}_{gps}(z^*) + \mathcal{H}_{lidar}(z^*) + \mathcal{H}_{pop}(z^*) + \mathcal{H}_{risk}(z^*) \tag{4.24}$$



(a) Success rates for BIT^*_{dist}.

(b) Success rates for PTP.

Figure 4.11: Planner success rates for BIT^*_{dist} and PTP over all altitudes.

and normalized using min-max normalization:

$$\mathcal{H}_{shift}(z^*) = \mathcal{H}_{total}(z^*) - \min(\mathcal{H}_{total}(z^*))J \tag{4.25}$$

$$\mathcal{H}_{norm}(z^*) = \frac{1}{\max(\mathcal{H}_{shift}(z^*))}\mathcal{H}_{shift}(z^*) \tag{4.26}$$

where J is a matrix of ones with the same dimensions as $\mathcal{H}_{total}$. To compare, we focus on daytime population for {20m, 60m} AGL flight and nighttime population for {122m, 600m} flight. Motion planners that found a solution are labeled on the top-left corner of each map.

For low-altitude flight (20m AGL) obstacle-related costs are prominent in $\mathcal{H}_{norm}$, where $\mathcal{H}_{norm} = 0$ is depicted in black with a gradient to white for $\mathcal{H}_{norm} = 1$ in Fig. 4.12. Manhattan, the Bronx, and portions of Queens/Brooklyn display high cost values attributed to tall buildings and urban canyon effects. At such a low altitude, a motion planner requires efficient obstacle-avoidance to find a feasible solution. As shown in Fig. 4.12a, for a long-range flight traversing through Manhattan only A^*_{dist} was able to find a solution. In contrast, for short-range flights over New Jersey, all

planners were able to generate a feasible flight path as shown in Fig. 4.12b. Fig. 4.12c shows a mid-range flight with some obstacles present over parts of Queens and Manhattan. The modest number of obstacles allowed three out of the five motion planners to terminate but with different path traits. As described below, A^*_{dist} followed a grid-based path that is minimum distance only with respect to that grid, while the BIT* variants took another option that is more direct because BIT* does not rely on the $5m$ resolution map grid apart from estimates of cost.

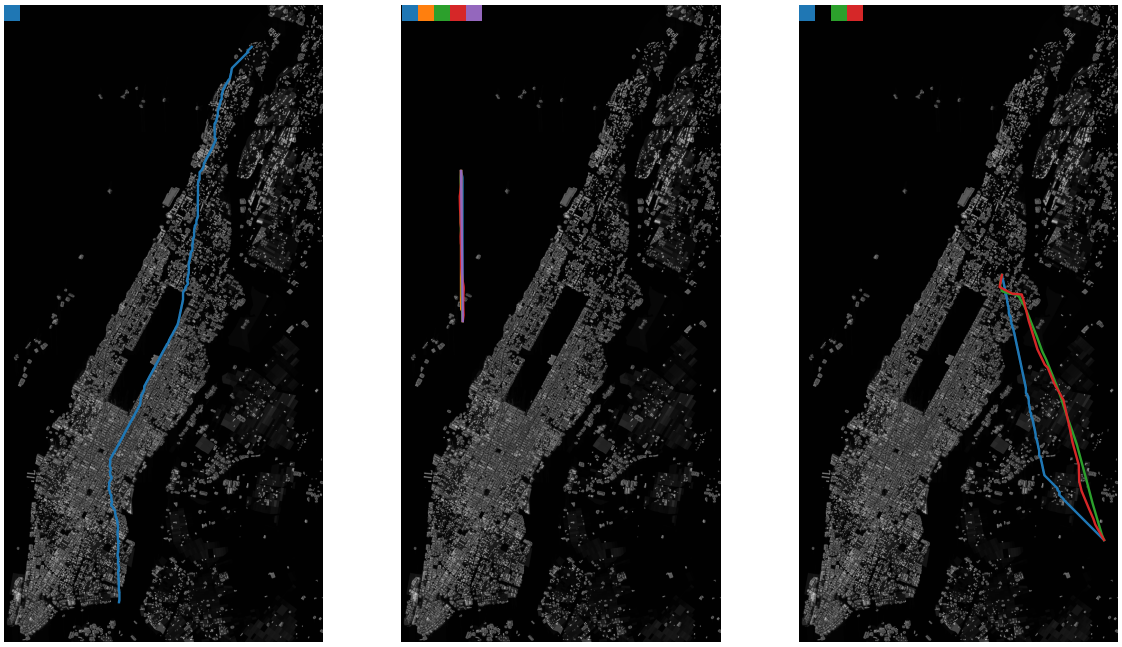


Figure 4.12: Example solution paths at 20m AGL, 5m resolution maps in New York City.

For mid-altitude flight (60m AGL), similar path and $\mathcal{H}_{norm}$ characteristics are observed in the Fig. 4.13 example paths. At this height, obstacles are only present in the Financial District (lower left) and Midtown Manhattan. Population now plays a more significant role in low-rise areas, especially the neighboring boroughs. Fig. 4.13a depicts a path attempting to traverse Midtown Manhattan. Motion planners circumvented the dense group of tall buildings with BIT^*_{dist} taking "shortcuts" to

minimize distance while BIT^*_{plus} navigates through lower population and risk areas. Fig. 4.13b investigates paths generated over the Hudson River. With no population or obstacle-related costs, all motion planners are capable of constructing feasible paths. BIT* variants and PTP take a direct approach from $\mathcal{Q}_S$ to $\mathcal{Q}_G$. The A* variants follow eight-connected grids. With the $5m$ resolution case study map, each A* step is either $5m$ along a primary compass direction or $7.07m$ along a 45 degree diagonal. This grid-based routing process leads to longer thus higher cost paths compared with direct routes, e.g., a distance cost of $5625m$ for PTP versus $6092m$ for A^*_{dist} in the example from Fig. 4.13b. This phenomenon is also observed in Fig. 4.13c.

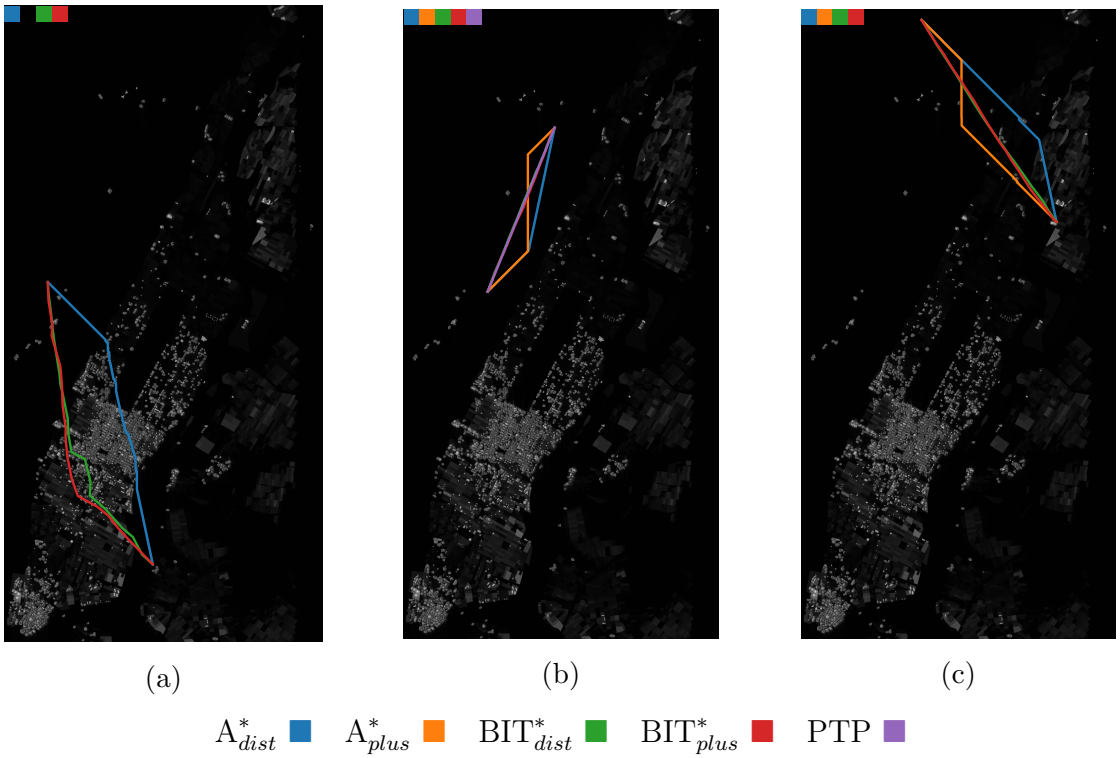


Figure 4.13: Example solution paths at 60m AGL, 5m resolution maps in New York City.

For high-altitude flight (122m AGL), tall buildings only remain in highly concentrated areas of the Financial District and Midtown Manhattan. Fig. 4.14a and Fig. 4.14b illustrate the success of motion planners when flying in these areas for short and long-range flight. In the first case, paths are generated from New Jersey, across

the Hudson, and into Midtown Manhattan. Given the long range and abundance of obstacles upon approach, only A^*_{dist} and the BIT* variants successfully terminated. However, with a reduced distance between $\mathcal{Q}_S$ and $\mathcal{Q}_G$, A^*_{plus} now terminates and takes a safer path than the rest. Furthermore, range can also be an issue for BIT^*_{plus}. As shown in Fig. 4.14c, BIT^*_{dist} and BIT^*_{plus} generate noticeably different paths. Given BIT^*_{plus} had to search more nodes to minimize non-distance costs, it had fewer batches, or iterations, to return its best-cost solution by the planning deadline.

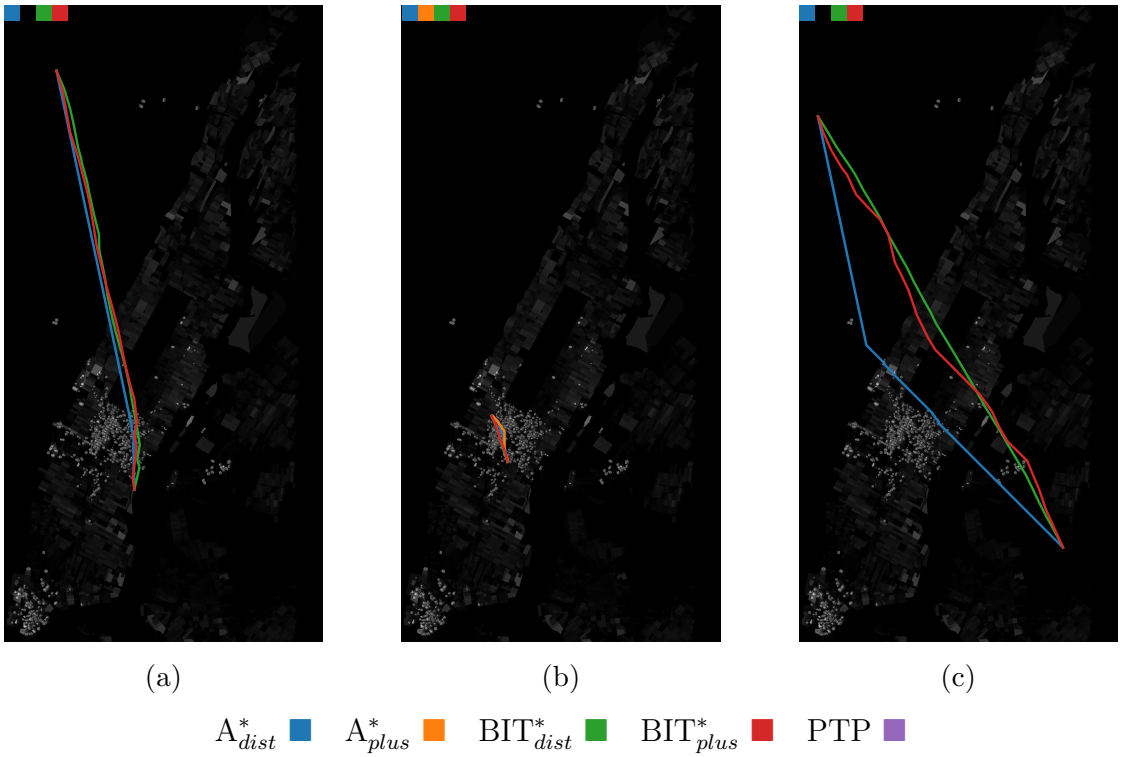


Figure 4.14: Example solution paths at 122m AGL, 5m resolution maps in New York City.

Above all buildings at 600m AGL, only distance and population remain as non-trivial costs. As shown in Fig. 4.15a, lack of obstacles and short travel distance is ideal for all planners. However, this may not be the case as range increases per Figs. 4.15c and 4.15c. Along the Hudson River, distance is the only cost to optimize, making PTP the best motion planner in this example. However, upon entering Manhattan, BIT^*_{plus} becomes more suitable as it selects a route over lower population

areas. The distance-population tradeoff demonstrates the benefits of geometric versus sampling-based planners. Collectively, these case studies illustrate the pros and cons of each planner thus motivate motion planning algorithm selection.

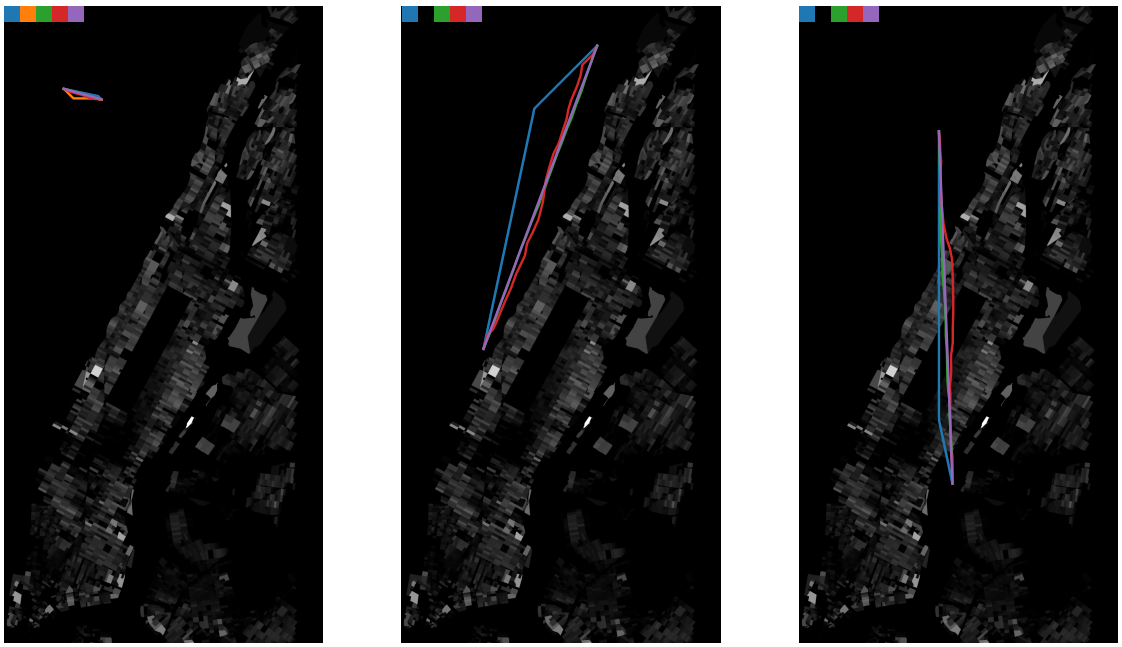


Figure 4.15: Example solution paths at 600m AGL, 5m resolution maps in New York City.

4.5 Decision Trees for Motion Planning ASP

Data-driven ASP decision trees were generated for two selection criteria: (1) Total cost and (2) Planner success. For each altitude, planning instance ranges were divided into nine intervals. The 36 (altitude, range) pairs were applied to planning instances as follows:

$$\hat{\mathcal{P}}_{z,r_{min},r_{max}} = \{p \in \mathcal{P} \mid \mathcal{Q}_{S,z} = z, r_{min} < d_{euc}(\mathcal{Q}_S, \mathcal{Q}_G) \leq r_{max}\} \tag{4.27}$$

where z is aircraft AGL altitude and $[r_{min}, r_{max}]$ describes the range interval for that bin. Each planner result is matched to a bin $\hat{\mathcal{P}}_{z,r_{min},r_{max}}$, and normalized mean cost and success rate were calculated as follows:

$$\mathcal{T}_{cost}(a, z, r_{min}, r_{max}) = \text{mean}\left(\left\{ \frac{cost(a,p)}{c_{max}} \;\middle|\; p \in \hat{\mathcal{P}}_{z,r_{min},r_{max}} \right\}\right) \tag{4.28}$$

$$\mathcal{T}_{success}(a, z, r_{min}, r_{max}) = sr(a, \hat{\mathcal{P}}_{z,r_{min},r_{max}}) \tag{4.29}$$

where $cost(\cdot)$ returns the Monte Carlo weighted cost for the algorithm as in Eq. 4.21, planning instance pair (a, p) and $c_{max} = \max(\{cost(a,p) \mid a \in \mathcal{A}, p \in \hat{\mathcal{P}}_{z,r_{min},r_{max}}\})$.

Each decision tree branch selects the planner with the minimum cost or maximum success rate such that:

$$\mathcal{T}_{C,i,j} = \text{argmin}(\{\mathcal{T}_{cost}(a, z, r_{min}, r_{max}) \mid a \in \mathcal{A}\}) \tag{4.30}$$

$$\mathcal{T}_{S,i,j} = \text{argmax}(\{\mathcal{T}_{success}(a, z, r_{min}, r_{max}) \mid a \in \mathcal{A}\}) \tag{4.31}$$

where $\mathcal{T}_{C,i,j}$ and $\mathcal{T}_{S,i,j}$ represent cost and success based selections for the ith altitude and jth range bin, respectively. Fig. 4.16 and Fig. 4.17 show these results as heatmaps.

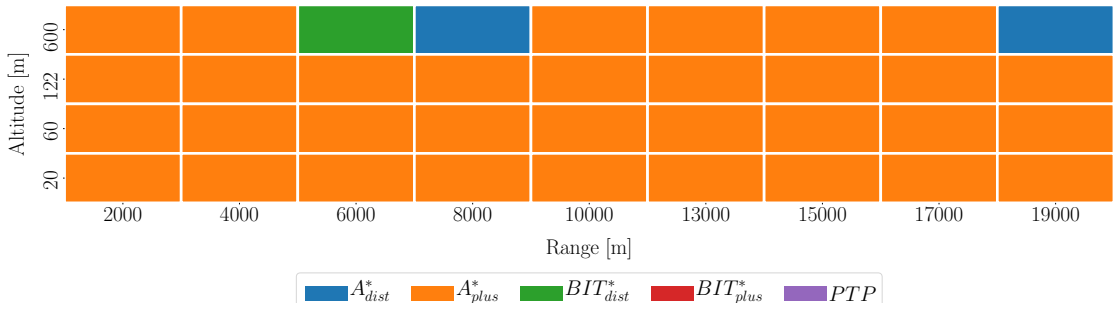


Figure 4.16: Cost decision tree heat map.

Decision trees were formulated for each heatmap as shown in Fig. 4.18. Excluding the root node, cost-based decision tree T_C contained seven decision nodes with six leaf

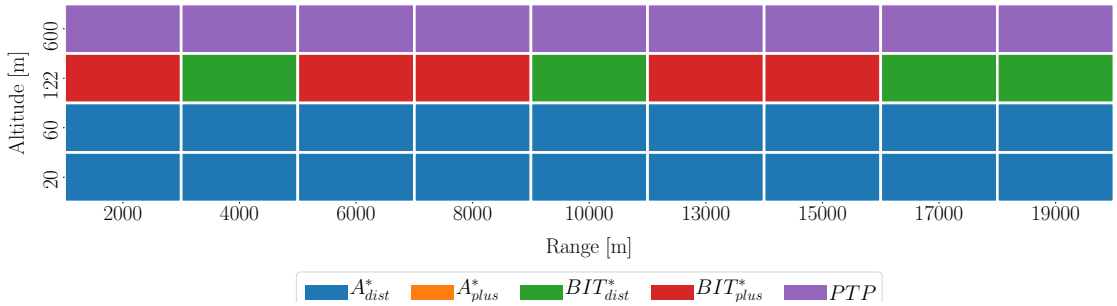
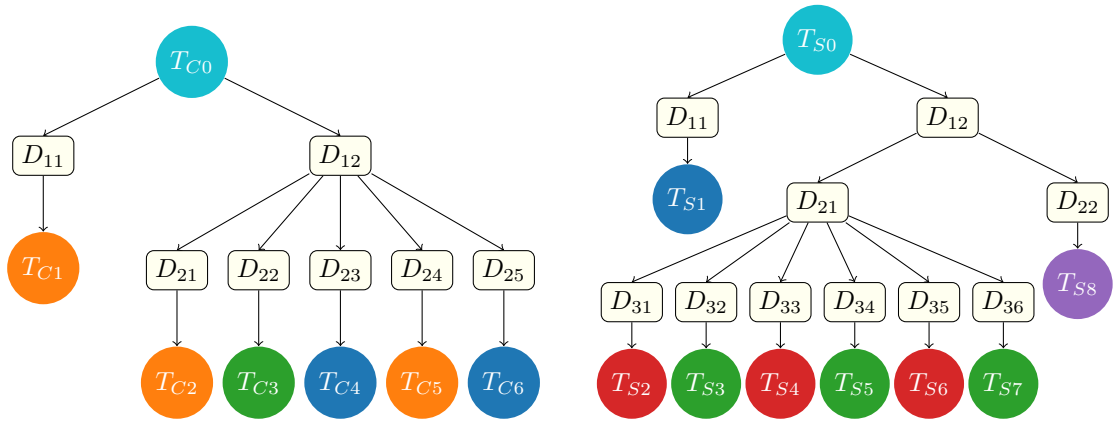


Figure 4.17: Success decision tree heat map.

states as labeled in Tables 4.2 and 4.3, respectively. Success-based decision tree T_S contained ten decision nodes and eight leaf states per Tables 4.4 and 4.5, respectively. Decision nodes indicate the altitude or range condition met by child states; leaf states represent the final algorithm selection, color-coded for readability.



(a) Cost-based decision tree T_C.

(b) Success-based decition tree T_S.

Figure 4.18: Algorithm selection decision trees generated using cost and success rate planning benchmarks derived from Monte Carlo simulations.

For both decision trees, altitude had the most significant effect. An altitude in $\{20m, 60m, 122m\}$ for T_C resulted in A^*_{plus} being the most effective planner due to its cost-optimizing behavior. At ceiling altitude 600m AGL, A^*_{dist} and BIT^*_{dist} had a more prominent role since only m_{dist} and m_{pop} are relevant. As cost terms diminish, the benefit of cost-optimal planners is reduced.

In contrast, for T_S altitudes of 20m or 60m indicate A^*_{dist} as the planner most

Table 4.2: Cost-based tree decisions.

Label	Type	Definition
D_{11}	Decision	$alt \in \{20, 60, 122\}$
D_{12}	Decision	$alt \in \{600\}$
D_{21}	Decision	$range \in [1000, 5000]$
D_{22}	Decision	$range \in (5000, 7000]$
D_{23}	Decision	$range \in (7000, 9000]$
D_{24}	Decision	$range \in (9000, 18000]$
D_{25}	Decision	$range \in (18000, 20000]$

Table 4.3: Cost-based tree state.

Label	Type	Definition
T_{S0}	State	Root
T_{S1}	State	A^*_{plus}
T_{S2}	State	A^*_{plus}
T_{S3}	State	BIT^*_{dist}
T_{S4}	State	A^*_{dist}
T_{S5}	State	A^*_{plus}
T_{S6}	State	A^*_{dist}

Table 4.4: Success-based tree decisions.

Label	Type	Definition
D_{11}	Decision	$alt \in \{20, 60\}$
D_{12}	Decision	$alt \in \{122, 600\}$
D_{21}	Decision	$alt = 122$
D_{22}	Decision	$alt = 600$
D_{31}	Decision	$range \in [1000, 3000]$
D_{32}	Decision	$range \in (3000, 5000]$
D_{33}	Decision	$range \in (5000, 9000]$
D_{34}	Decision	$range \in (9000, 12000]$
D_{35}	Decision	$range \in 12000, 16000]$
D_{36}	Decision	$range \in (16000, 20000]$

Table 4.5: Success-based tree state.

Label	Type	Definition
T_{S0}	State	Root
T_{S1}	State	A^*_{dist}
T_{S2}	State	BIT^*_{plus}
T_{S3}	State	BIT^*_{dist}
T_{S4}	State	BIT^*_{plus}
T_{S5}	State	BIT^*_{dist}
T_{S6}	State	BIT^*_{plus}
T_{S7}	State	BIT^*_{dist}
T_{S8}	State	PTP

likely to find a solution since it navigates obstacles efficiently. At high-altitude 122m AGL flight, BIT^*_{dist} and BIT^*_{plus} become interchangeable. Their sampling-based path construction allows them to find paths more often without being impacted by map resolution as much as their grid-based counterparts. Above all buildings, the geometric-based planner PTP always finds a solution thus is preferred.

4.6 Neural Networks for Motion Planning ASP

This section describes a neural network approach to the motion planning ASP. Training and test data are generated by executing all motion planners in $\mathcal{A}$ for all problem instances. Problem instances $\mathcal{P}$ are randomly generated over a prescribed region described by maps $\mathcal{H}$ with start state $\mathcal{Q}_S$ and goal state $\mathcal{Q}_G$. For each algorithm-problem instance pair (a, p), a score is assigned from cost weighting scheme $\mathcal{W}$ applied

over metrics m_p and m_m, defined in Ch. III, summed over the generated path. Each ASP selection output $\mathcal{S}$ is computed from network inputs $\boldsymbol{x}_{in}$ composed from planning problem values $\{\mathcal{Q}_S, \mathcal{Q}_G, \mathcal{W}, \eta\}$. η, defined below, is a latent representation of all feature maps in $\mathcal{H}$ constructed by an encoding function $\mathcal{E}$.

4.6.1 Metric Encoding

Raw cost map data are not practical as direct ASP network inputs because the remaining input data, e.g., $\mathcal{Q}_S, \mathcal{Q}_G$, and $\mathcal{W}$, is comparably important but smaller in quantity. We therefore convert each cost map into a lower-dimensional latent space comparable in size to remaining network inputs. Using $\mathcal{Q}_S, \mathcal{Q}_G \in p$ an axis-aligned bounding box (AABB) $\hat{\mathcal{B}}$ submap is generated for each planning instance p such that:

$$\hat{\mathcal{B}} = \begin{pmatrix} \min(Q_{S,x}, Q_{G,x}) - \delta b \\ \min(Q_{S,y}, Q_{G,y}) - \delta b \\ \max(Q_{S,x}, Q_{G,x}) + \delta b \\ \max(Q_{S,y}, Q_{G,y}) + \delta b \end{pmatrix} \tag{4.32}$$

where $Q_{S,x}, Q_{S,y}, Q_{G,x}$, and, $Q_{G,y}$ are the x, y coordinates of the start and goal configurations in the inertial frame, respectively, and δb is added or subtracted to enlarge the bounding box by a specified distance buffer.

Latent space representation η of a given map $\mathcal{H}$ is constructed from descriptive statistics. Recall from Ch. III that we have defined five maps: obstacle $\mathcal{H}_{obs}$, GPS $\mathcal{H}_{gps}$, Lidar $\mathcal{H}_{lidar}$, population $\mathcal{H}_{pop}$, and risk $\mathcal{H}_{risk}$. *min* (minimum), *max* (maximum), and *med* (median) capture the range of values each metric takes. *mean* and *std* quantify the metric's average value and standard deviation, respectively. *skew* (skewness) and *kurt* (kurtosis) [111] summarize the metric's statistical symmetry or lack thereof. Latent space representation η for an obstacle or metric map $\mathcal{H}$ is thus

defined by:

$$\eta(\mathcal{H}) = \Big(\min(\mathcal{H}), \max(\mathcal{H}), \text{med}(\mathcal{H}), \text{mean}(\mathcal{H}), \text{std}(\mathcal{H}), \text{skew}(\mathcal{H}), \text{kurt}(\mathcal{H})\Big) \quad (4.33)$$

4.6.2 Neural Network Designs

Two types of motion planning ASP neural networks were constructed: (1) Cost and success networks for each individual planner $a \in \mathcal{A}$ later used in a hybrid seletion N_H scheme as shown in 4.19, and (2) A single unified network N_U predicting the best candidate motion planner $a* \in \mathcal{A}$ directly.

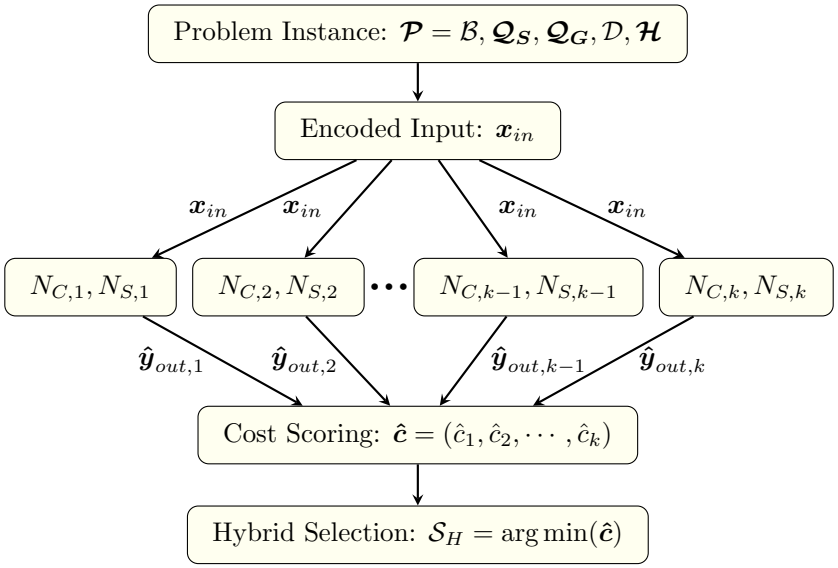


Figure 4.19: ASP over networks N_i estimate motion planner i scores $\hat{c}_i$.

To construct network input matrix $\boldsymbol{x}_{in}$, maps $\boldsymbol{\mathcal{H}}$ in p are encoded into η such that $\eta = \{\eta_{\mathcal{H}} \mid \forall \mathcal{H} \in \boldsymbol{\mathcal{H}}\}$, where each $\eta(\mathcal{H})$ is a row of network inputs. Matching weight vector $w \in \mathcal{W}$ is appended to η in a new column. Two additional columns input UAS above-ground height z^* and map resolution r^* as values pertinent to all cost maps. Hence, network input is a matrix of size $(k,\, d + 3)$ for k maps and d latent statistical

features:

$$\boldsymbol{x}_{in} = \begin{pmatrix} \eta_1 & w^{(1)} & z^* & r^* \\ \vdots & \vdots & \vdots & \vdots \\ \eta_{k-1} & w^{(k-1)} & z^* & r^* \\ \eta_k & w^{(k)} & z^* & r^* \end{pmatrix} \tag{4.34}$$

For our networks, $k = 5$ maps and $d = 7$ latent statistics. Each network output is then concatenated as follows:

$$\hat{\boldsymbol{y}}_{out,i} = \begin{pmatrix} \hat{y}_{C,i} \\ \hat{y}_{S,i} \end{pmatrix} \tag{4.35}$$

where $\hat{y}_{C,i}$ is the estimated cost of the ith planner output by cost network $N_{C,i}$, and $\hat{y}_{S,i}$ is a binary prediction of ith planner success output by success network $N_{S,i}$. Network output $\hat{y}_{S,i}$ is binary to match our defined success function S_i in Eq. 4.23. Overall planner i score $\hat{c}_i$ is then computed from network outputs $\hat{\boldsymbol{y}}_{out}$:

$$\hat{c}_i = \begin{cases} \hat{y}_{C,i} & \text{if } \hat{y}_{S,i} = success \\ 1 & \text{otherwise} \end{cases} \tag{4.36}$$

The final ASP output $\mathcal{S}_H$ has the lowest predicted cost $\hat{c}_i \in \hat{\mathbf{c}}$.

A "unified" network with final ASP output $\mathcal{S}_U$ was also developed. As depicted in Fig. 4.20, a single neural network $\mathcal{N}_U$ identifies the best algorithm $a^* \in \mathcal{A}$ for a planning instance using the same encoded input matrix $\boldsymbol{x}_{in}$. Encoder network $\mathcal{N}_U$ outputs an ordered categorical probability distribution indicating each of k planning algorithms' selection likelihood $\hat{\boldsymbol{\rho}} = (\hat{\rho}_1, \hat{\rho}_2, \cdots, \hat{\rho}_k)$. The final selection is made by choosing the algorithm with the highest selection likelihood per Fig. 4.20.

Neural networks were implemented using the Keras[†] deep learning API on an i9-

[†]https://keras.io/

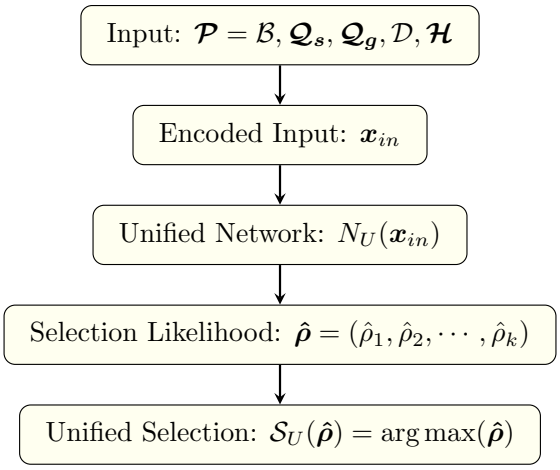


Figure 4.20: Unified ASP strategy $\mathcal{S}_U$ predicted by a singular neural network $\mathcal{N}_U$.

9900K CPU @ 3.60GHz with a GeForce RTX 2080 Ti GPU for training. Monte Carlo simulations were separated at random for training and validation using a 50-50 split, 30000 each, beyond which more training data was unnecessary. Individual networks $N_{S,i}$ and a unified N_U categorical network were each composed of three layers (input, hidden, output) with activation functions relu, relu, and softmax respectively, where *relu* is a rectified linear unit. Similarly, regression cost estimators $N_{C,i}$ were constructed as three layer networks with relu activation at all layers. All networks were fully-connected with the following number of nodes or units in their (input, hidden, output) layers: $N_{S,i}$: (8, 1, 2), $N_{C,i}$: (24, 8, 1), N_U: (16, 8, 6). Unit counts were selected to avoid overfitting while offering sufficient complexity to capture selection model differences.

4.7 ASP Results

This section presents results for decision tree and neural network ASP strategies. First, a ground truth analysis is conducted over the Monte Carlo simulations used for ASP validation. Each problem instance is matched to a "best" planner based on ground truth data, and a sensitivity analysis for map-based metrics defined in Ch. III is presented. Cost and success rate performance benchmarks are presented for

the individual planner neural networks used in hybrid strategy N_H. ASP results are examined for the cost-based T_C and success-based T_S decision trees and compared with hybrid N_H and unified N_U neural network ASP results.

4.7.1 Ground Truth

Table 4.6 summarizes results from the validation dataset. At the lowest altitude, the A* variants are most likely to be selected. This coincides with previous evidence of their better performance in obstacle-dense environments. As altitude increases, BIT* variants and PTP are selected more often, thriving in the ence of tall buildings.This trend continues up to ceiling-altitude flight, where PTP becomes the best performing planner since flight occurs above all buildings. For 254 of the 30000 validation examples (less than 1 percent) all planners failed to find a solution, hence, so answer (N/A) is tallied. The majority of N/A cases occur at 20m AGL where either $\mathcal{Q}_S$ or $\mathcal{Q}_G$ might be randomly sampled in an enclosed space, for example. At higher altitudes, long-range paths through tall buildings in Midtown Manhattan made it difficult for planners to find a solution due to collisions (BIT^*_{dist}, BIT^*_{plus}, PTP) or insufficient time for search (A^*_{dist}, A^*_{plus}).

Table 4.6: Ground truth motion planning ASP selection for validation data. Each of the four altitudes has 7500 validation cases.

Altitude	A^*_{dist}	A^*_{plus}	BIT^*_{dist}	BIT^*_{plus}	PTP	N/A
20	4215	1070	1218	355	436	217
60	1806	626	3005	720	1290	34
122	959	378	3114	624	2458	3
600	188	190	2903	457	3734	0
Total	7168	2264	10240	2156	7918	254

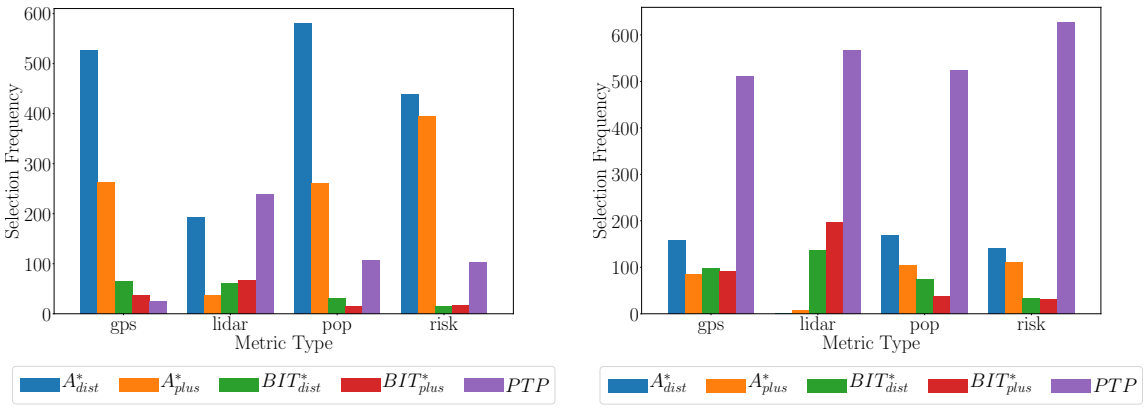
Table 4.7 summarizes the failure frequency for each planner at each altitude. By construction, the A* variants never define paths with obstacle conflict. Hence, A* failures are attributed to search spaces too large to navigate within the allotted time

frame (three minutes). A^*_{plus} was impacted most at lower altitudes while A^*_{dist} was affected uniformly. Likely, this is a result of non-distance metrics exhibiting greater variance at lower altitudes with less direct paths chosen. In contrast, BIT* and PTP failures are mainly the result of obstacle collisions. To generate paths, BIT* and PTP attempt to create collision-free lines between two given waypoints. For BIT* this is completed iteratively using random samples while PTP attempts to connect $\mathcal{Q}_S$ and $\mathcal{Q}_G$ directly. In dense obstacle areas, hence, at lower altitudes, these planners are more likely to fail as shown in Table 4.7. Above all buildings, PTP is guaranteed to find a solution.

Table 4.7: Number of planner failures over all Monte Carlo studies. Each of the four altitudes has 15000 total cases.

Altitude	A^*_{dist}	A^*_{plus}	BIT^*_{dist}	BIT^*_{plus}	PTP
20	619	11068	9841	9862	13175
60	704	10253	1590	1601	9725
122	780	9432	250	218	4973
600	818	8558	34	38	0

Sensitivity analysis for each map-based metric was performed on the ground truth validation dataset. The validation set was scraped to extract all planning instances where only a single map-based metric weight was non-zero. For low-altitude (20m AGL) and high-altitude (122m AGL) flight, selection frequency was tallied for each planner per metric type as shown in Fig. 4.21. For low-altitude flight, *gps*, *pop*, and *risk* have the least influence on algorithm selection, where the A* variants dominate. In contrast, for *lidar* there exists higher uncertainty in making a final selection as shown in Fig. 4.21a. At higher altitude flight, all map-based metrics become less relevant as they become uniform outside the urban canyon. This is depicted in Fig. 4.21b where PTP becomes the preferred motion planning candidate.



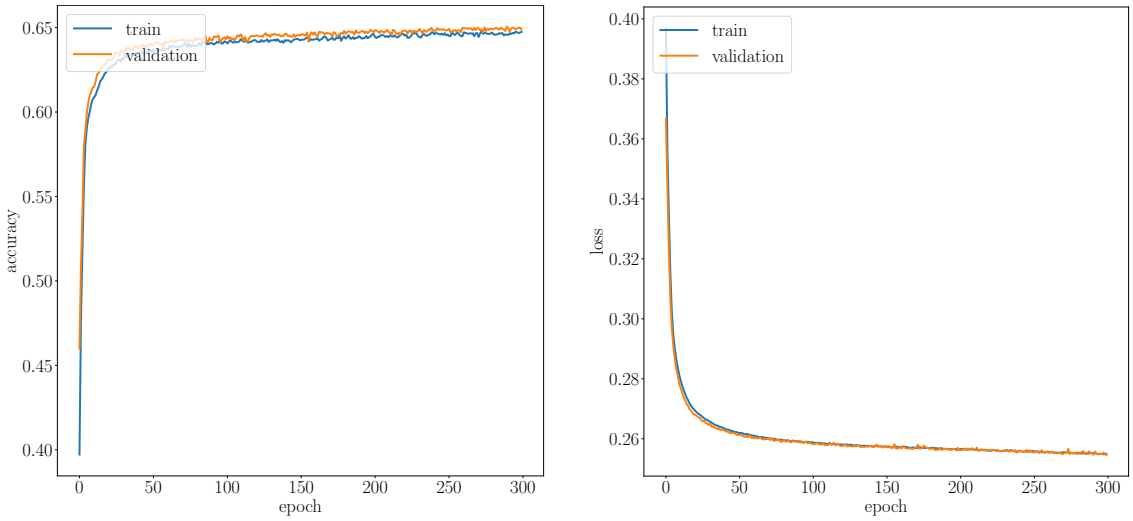
(a) Low-altitude ($20m$) selection frequency. (b) High-altitude ($122m$) selection frequency.

Figure 4.21: Ground truth ASP (best) planner selection histograms over the validation dataset for individual cost metrics.

4.7.2 Selection Results

Individual planner networks and the unified selection network architecture were each trained using the generated Monte Carlo cases, half used for training (30,000) and the rest (30,000) for validation. To avoid overtraining, with a batch size of 100, each network was terminated after 300 epochs. Overfitting was mitigated by using simple network architectures. As a demonstration, training accuracy and loss for the unified ASP network N_U are shown in Fig. 4.22. Training and validation both displayed smooth converging patterns expected of a good fit. Similar patterns were exhibited for all other trained networks.

Perfomance of the individual cost N_C and success N_S networks is summarized by Table 4.8. Cost-wise A^*_{plus}, BIT^*_{dist}, BIT^*_{plus}, and PTP had a mean absolute error (MAE) normalized average of 0.12 with standard deviation (STD) of 0.17 while A^*_{dist} performed significantly better with MAE and STD of 0.02 and 0.06, respectively. The underperformance of $N_{C,A^*_{plus}}$ and $N_{C,PTP}$ can likely be attributed to fewer successful (i.e., $\zeta \neq \emptyset$) training examples as indicated by the last table column showing success



(a) Unified network N_U training accuracy. (b) Unified network training loss.

Figure 4.22: Unified network N_U training performance over 300 epochs on Keras.

count out of 30000 total training examples. In contrast, the BIT* cost networks had sufficient successful training examples but their sampling-induced errors were appreciable. The A^*_{dist} cost network had the best performance as it did not suffer from the aforementioned issues.

Table 4.8: Individual cost and success network performance benchmarks.

Planner	N_C MAE	N_C STD	N_S ACC	Count
A^*_{dist}	0.02	0.06	0.99	28527
A^*_{plus}	0.13	0.17	0.89	10315
BIT^*_{dist}	0.10	0.16	0.93	24120
BIT^*_{plus}	0.11	0.17	0.93	24159
PTP	0.13	0.18	0.89	16048

Success networks N_S demonstrate good accuracies (ACC) regardless of the planner. Similar to the previous analysis, $N_{S,A^*_{dist}}$ had the highest likelihood of predicting the success of its planner with 99 percent accuracy. Despite $N_{S,A^*_{plus}}$ and $N_{S,PTP}$ having the worst success prediction accuracies at 89 percent, the success networks performed reasonably. Cost networks N_C alone may not reliably select the best performing planner due to their substantial errors, but augmentation with improved N_S

achieves promising results.

Overall performance of the decision trees from Sec. 4.5 and neural networks from Sec. 4.6 is summarized in Table 4.9. The rightmost column indicates the number of validation set cases for which the planner in that row is best per Table 4.6. The cost T_C and success T_S decision trees were the worst performing ASP candidates with average algorithm selection accuracies of 9 and 23 percent, respectively. Both decision trees performed poorly with the BIT* variants as well as PTP. T_C performs well with A^*_{plus}, but the decision trees' overall performance indicates a more sophisticated ASP scheme is needed.

Table 4.9: Comparison of decision tree and neural network accuracies.

Planner	T_C	T_S	N_U	N_H	Count
A^*_{dist}	0.01	0.49	0.67	0.83	7168
A^*_{plus}	0.90	0.00	0.49	0.72	2264
BIT^*_{dist}	0.05	0.10	0.82	0.79	10240
BIT^*_{plus}	0.00	0.15	0.00	0.74	2156
PTP	0.00	0.26	0.70	0.68	7918
Weighted Accuracy	0.09	0.23	0.67	0.76	29746

The second-best ASP technique was unified neural network N_U. Significant improvements were made for all planning algorithms except the plus variants. A^*_{plus} was selected with less than 50 percent accuracy and BIT^*_{plus} was never selected. BIT^*_{dist} selection performed best with 83 percent accuracy, while A^*_{dist} and PTP selection had satisfactory results with 67 percent and 70 percent accuracy, respectively. Combining these results, the unified ASP NN technique had a 2.5x improvement over the decision trees with an overall accuracy of 67 percent. This result demonstrates the learning advantages of using a more-informed neural network over a simpler decision tree classifier.

The hybrid ASP technique N_H had the best performance with an overall accuracy of 77 percent, a 3x improvement over decision tree T_C and T_S selection accuracy. A^*_{dist}

and A^*_{plus} selection accuracy improved by 16 and 23 points respectively with BIT^*_{plus} having the most significant jump at 74 points. While BIT^*_{plus} and PTP selection accuracies were not as high, overall performance was satisfactory for this first work in learning an ASP function for motion planning. These results show neural networks, particularly a hybrid formulation, are a promising approach to the motion planning algorithm selection problem.

4.8 Conclusion

This chapter has presented an in-depth analysis of small UAS urban flight planning and ASP over geometric (PTP), graph-based (A^*_{dist}, A^*_{plus}), and sampling-based (BIT^*_{dist}, BIT^*_{plus}) motion planners, along with a novel heuristic h_{plus}. Path cost, execution time, and success rate benchmarks were calculated using Monte Carlo simulations. PTP was the fastest algorithm for all planning instances and had the highest success rate when with flight paths strictly above buildings. A^*_{dist} improved upon this result with comparable execution times and the best overall success rate. However, its distance-only cost function failed to capture the diversity of the proposed multi-objective cost map. In contrast, metric map based A^*_{plus} produced the lowest cost paths but often took too long to complete within our three-minute planning time constraint. The BIT* variants shared similar path costs; BIT^*_{dist} had significantly lower execution times, making less superfluous metric checks than BIT^*_{plus}.

Future work can expand these benchmarks to consider additional motion planners as well as different computing platforms. Dubins curves [54] can extend the presented work to fixed-wing aircraft and steered ground robots. Less grid-stringent planners such as Theta* [99] or multi-resolution motion planners may improve the success rate of A^*_{plus} when planning on high resolution maps. Work on adaptively informed trees (AIT*) [112], the successor to BIT*, was published during the writing of this dissertation. The addition of AIT* to the algorithm portfolio would maintain BIT*'s

optimality while likely improving time-to-convergence.

The second half of this chapter addressed motion planning ASP. Rule-based and neural network selection frameworks were presented and analyzed. Rule-based decision trees constructed from Monte Carlo simulations were unable to reliably select the best algorithm, resulting in less than 25 percent selection accuracy. Alternative *hybrid* and *unified* neural network-based approaches achieved 2.5x and 3x accuracy improvements over decision tree results, respectively. The neural networks were able to capture more information than in our simple decision tree rules. For greater detail, raw map data can serve as selection inputs and may improve accuracy. However, this dissertation's latent map representations can be applied directly to multiple urban environments in future work.

CHAPTER V

Conclusion and Future Work

This dissertation investigates small UAS real-time motion planning. Fail-safe strategies for urgent landing planning enable a UAS to safely land despite anomalies or failures. To avoid rigidity, the motion planning algorithm selection problem (ASP) is defined for small UAS applications. Emphasis in this research is placed on comprehensive ASP metric definition cutting across a variety of aerial ASP applications. Machine learning approaches to ASP are investigated for small multicopter urban flight over a variety of flight altitudes and environmental complexities.

Fail-safe studies served as a precursor for defining the motion planning algorithm selection problem (ASP). Improving on contemporary "fly-home" or automatic landing protocols, three alternative data-driven fail-safe protocols were presented. However, these strategies alone were insufficient due to their reliance on specific user-selected motion planning algorithms. While suitable for short-range case studies, the selected algorithms would face a computational burden for longer flights. To eliminate user selection bias and investigate a representative pool of motion planners, additional work followed to formulate a better solution for the motion planning ASP.

Metric maps were constructed to provide insights into the diversity of the urban canyon needed to construct safe and efficient flight plans. GPS and Lidar maps demonstrated the complementary nature of these navigation sensors in contrasting

obstacle densities. Computed a priori, these maps can predict each sensor's value subject to geolocation and flight altitude. A daytime vs. nighttime work-week population model was presented; future work should extend this model to higher-resolution 24-hour data to reduce risk to overflown population at all times. Besides proximity risk, additional research is needed to model system, actuator, sensor, and weather-related risks encountered by small UAS, as summarized in Table 5.1, for robust guidance, navigation, and control (GNC). Analogous analysis in other major cities and extensions of presented metrics to full 3D cost maps are next steps towards modeling a more realistic urban flight planning environment.

Table 5.1: Common risks encountered by small UAS.

Type	Description	Examples
System	A hardware or software failure resulting in a system freeze, coding error, reboot, component failure, or a complete shutdown.	Deadlock [113, 114], overheating [115], electrical shorts [116], software risks [117]
Actuators	Control surfaces are irresponsive or fail to reach a target configuration given a threshold.	Shaft failures [118], PWM relay errors [119], pneumatic/hydraulic faults [120]
Sensors	Onboard sensing tools provide inaccurate representations of the world around them.	Faulty sensors, obstructed view, drifting sensor readings, urban canyon effects
Weather	Hazardous climate conditions influencing system sensing and/or performance.	Cold impact on batteries [121], poor visibility, snow/ice buildup, turbulent winds [122]

To the author's best knowledge, the motion planning ASP was first addressed in this dissertation. Rule-based and neural network selection frameworks were presented and analyzed. Rule-based decision trees were simple to construct but unable to capture both complex cost metrics and algorithm execution properties. The two investigated neural network based ASP formulations produced promising results, with a hybrid two-stage selection scheme having the best performance with an overall 75 percent selection accuracy. Although this work provides a baseline, additional work will

likely improve this accuracy figure by exploring altered metrics, network inputs, and architectures. Using raw map data in convolutional neural network (CNN) modules may identify patterns lost in latent feature conversions. If deployed in a real-world platform, a comparison of online algorithm selection and concurrent motion planning will be required to analyze a configurable hardware-software suite's execution trade-offs. While this work focused on motion planning, a similar investigation may be performed to tackle the sensor selection problem (SSP) and other GNC algorithms, e.g., controllers, state estimators, and signal filters.

APPENDICES

APPENDIX A

Datasets

The following appendix describes the online datasets used in this dissertation.

PLUTO: The Primary Land Use Tax Lot Output (PLUTO) data file contains extensive land use and geographic data for New York city. The PLUTO dataset is created by the NYC Department of City Planning as part of its Open Data Initiative. Data is available to download as an ASCII Comma-delimited File (CSV) with over seventy fields/attritutes for each NYC tax lot.

Relevant Fields:

- BOROUGH: The borough in which the tax lot is located.
- TAX BLOCK: Number of tax block in which tax lot is located.
- TAX LOT: Unique number designation within a tax block.
- LAND USE: Used to identify tax lots with buildings.
- NUM FLOORS: Number of stories for tallest building on tax lot.

Link: `https://www1.nyc.gov/site/planning/data-maps/open-data/dwn-pluto-mappluto.page`

MapPLUTO: MapPLUTO is a complementary database to be used in tandem with PLUTO. MapPLUTO matches tax lot geographic features dervied from the NYC Depart of Finance's Digital Tax Map (DTM). Tax lot polygons are available

with shorelines clipped or water included. Polygon outines are provided as latitude and longitude using the WGS84 CRS. Data can be downloaded as a File Geodatabse (FGDB, proprietary) or Shapefile (SHP).

Link: `https://www1.nyc.gov/site/planning/data-maps/open-data/dwn-pluto-mappluto.page`

OpenStreetMap: OpenStreetMap (OSM) is an opensource project founded by Steve Coast in 2004. Similar to Wikipedia, worldwide users aggregate and vet data from manual surveys, GPS devices, aerial photography, and other GIS sources along with their local knowledge of an area. All crowsourced data is free to share, modify, and use under the Open Database License (ODbL). OSM WGS84 data can be downloaded as an Extensible Markup Language File (XML) for a given bounding box or the entire OSM database is available from Planet OSM.

OSM attributed-tagged data falls into one of three categories:

- Nodes: Points representing features without a size. Standalone nodes can include park benches or traffic signals.
- Ways: Lists of nodes forming polylines or polygons if they form a closed loop. Typical open ways include rivers and roads. Closed ways, or polygons, are used to define important physical areas, especially buildings.
- Relations: Collections of nodes, ways, and other relations. Most often used to group multiple ways part of a common collection, such as a university.

Link: `https://www.openstreetmap.org/`

TIGER: Topologically Integrated Geographic Encoding and Referencing (TIGER) is a format used by the United States Census Bureau to define land-based features across the nation. A special population-specific TIGER database is accessible at the start of every decade for all fifty states and the District of Columbia. At this time data is not available for Puerto Rico or the Island areas. Population and housing unit count estimates are provided for each census block as defined by the Census Bureau. Data after 2007 is available to download as SHP files and CSV files before that.

Link: `https://www.census.gov/geographies/mapping-files.html`

CelesTrak: CelesTrak is a repository used to keep track of all human-made objects, or satellites, in Earth's orbit. The Two-Line Element (TLE) format is used to identify all satellites added by the North American Aerospace Defense Command (NORAD). In the 1980s, the United States Department of Defense released the equations and source code used to predict satellite orbits. Changes made to the original code were not released, however, independent efforts, technical papers, and source code have kept a non-proprietary version up-to-date.

Link: `https://celestrak.com/`

APPENDIX B

Software

Skyfield: Skyfield is a Python module used to compute positions of stars, planets, and satellites in orbit around the Earth. Using loaded TLE data, objects of interest are run through the SGP4 satellite propagation routine with Python's NumPy numerical library as its only binary dependency, used for computational efficiency.

Link: `https://rhodesmill.org/skyfield/`

Scikit-learn: Scikit-learn is a Python machine learning module simple predictive analysis. Built on top of Python's numerical and scientific libraries NumPy and SciPy, Scikit-learn features various classification, regression and clustering algorithms under the open source, commercially usable Berkeley Software Distribution (BSD) license.

Link: `https://scikit-learn.org/stable/index.html`

Keras: Keras is a Python deep learning API which runs on top of TensorFlow, an open source numerical computation machine learning library created by the Google Brain Team. Keras is built with fast deployment in mind, using prebuilt adaptive machine learning models and optimizers parameterized by the end user.

Link: `https://keras.io/`

BIBLIOGRAPHY

BIBLIOGRAPHY

[1] Lars Kotthoff. Algorithm Selection for Combinatorial Search Problems: A Survey. *AI Magazine*, 35(3):48, September 2014.

[2] Alec J. Ten Harmsel, Isaac J. Olson, and Ella M. Atkins. Emergency Flight Planning for an Energy-Constrained Multicopter. *Journal of Intelligent & Robotic Systems*, 85(1):145–165, January 2017.

[3] Jonathan D. Gammell, Siddhartha S. Srinivasa, and Timothy 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)*, pages 3067–3074, Seattle, WA, USA, May 2015. IEEE.

[4] Hamed Azami, Mohammad-Reza Mosavi, and Saeid Sanei. Classification of GPS Satellites Using Improved Back Propagation Training Algorithms. *Wireless Personal Communications*, 71(2):789–803, July 2013.

[5] Mitchell L. Moss and Carson Qing. The Dynamic Population of Manhattan. March 2012.

[6] John A. Volpe National Transportation Systems Center (U.S.). Unmanned Aircraft System (UAS) service demand 2015 - 2035 : literature review & projections of future usage, technical report, version 1.0 - February 2014. (DOT-VNTSC-DoD-13-01), February 2014.

[7] Evan Ackerman. Amazon to test delivery drone autonomy in the U.K. *IEEE Spectrum: Technology, Engineering, and Science News*, 2020.

[8] David P. Thipphavong, Rafael Apaza, Bryan Barmore, Vernol Battiste, Barbara Burian, Quang Dao, Michael Feary, Susie Go, Kenneth H. Goodrich, Jeffrey Homola, Husni R. Idris, Parimal H. Kopardekar, Joel B. Lachter, Natasha A. Neogi, Hok Kwan Ng, Rosa M. Oseguera-Lohr, Michael D. Patterson, and Savita A. Verma. Urban Air Mobility Airspace Integration Concepts and Considerations. In *2018 Aviation Technology, Integration, and Operations Conference*, Atlanta, Georgia, June 2018. American Institute of Aeronautics and Astronautics.

[9] Pedro F. Di Donato and Ella M. Atkins. An Off-Runway Emergency Landing Aid for a Small Aircraft Experiencing Loss of Thrust. In *AIAA Infotech @*

Aerospace, Kissimmee, Florida, January 2015. American Institute of Aeronautics and Astronautics.

[10] Pedro F. A. Di Donato and Ella M. Atkins. Evaluating Risk to People and Property for Aircraft Emergency Landing Planning. *Journal of Aerospace Information Systems*, 14(5):259–278, May 2017.

[11] Jeremy Castagno, Cosme Ochoa, and Ella Atkins. Comprehensive Risk-based Planning for Small Unmanned Aircraft System Rooftop Landing. In *2018 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 1031–1040, Dallas, TX, June 2018. IEEE.

[12] Maren Bennewitz, Wolfram Burgard, and Sebastian Thrun. Finding and optimizing solvable priority schemes for decoupled path planning techniques for teams of mobile robots. *Robotics and Autonomous Systems*, 41(2-3):89–99, November 2002.

[13] Mitul Saha and Pekka Isto. Multi-Robot Motion Planning by Incremental Coordination. In *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5960–5963, Beijing, China, October 2006. IEEE.

[14] J. van den Berg, J. Snoeyink, M. Lin, and D. Manocha. Centralized path planning for multiple robots: Optimal decoupling into sequential plans. In *Robotics: Science and Systems V*. Robotics: Science and Systems Foundation, June 2009.

[15] Kamal Kant and Steven W. Zucker. Toward Efficient Trajectory Planning: The Path-Velocity Decomposition. *The International Journal of Robotics Research*, 5(3):72–89, September 1986.

[16] David Hsu, Robert Kindel, Jean-Claude Latombe, and Stephen Rock. Randomized Kinodynamic Motion Planning with Moving Obstacles. *The International Journal of Robotics Research*, 21(3):233–255, March 2002.

[17] Eitan Marder-Eppstein, Eric Berger, Tully Foote, Brian Gerkey, and Kurt Konolige. The Office Marathon: Robust navigation in an indoor office environment. In *2010 IEEE International Conference on Robotics and Automation*, pages 300–307, Anchorage, AK, May 2010. IEEE.

[18] Eugene Nudelman, Alex Devkar, Yoav Shoham, Kevin Leyton-Brown, and Holger Hoos. SATzilla: An algorithm portfolio for SAT. May 2004.

[19] L. Xu, F. Hutter, H. H. Hoos, and K. Leyton-Brown. SATzilla: Portfolio-based Algorithm Selection for SAT. *Journal of Artificial Intelligence Research*, 32:565–606, July 2008.

[20] Bryan Silverthorn and Risto Miikkulainen. Latent class models for algorithm portfolio methods. In *Proceedings of the twenty-fourth AAAI conference on artificial intelligence*, AAAI'10, pages 167–172. AAAI Press, 2010.

[21] Adrian Balint, Anton Belov, Daniel Diepold, Simon Gerber, Matti Järvisalo, and Carsten Sinz, editors. *Proceedings of SAT Challenge 2012: Solver and benchmark descriptions*, volume B-2012-2 of *Department of computer science series of publications b*. University of Helsinki, 2012.

[22] Yuri Malitsky, Ashish Sabharwal, Horst Samulowitz, and Meinolf Sellmann. Algorithm Portfolios Based on Cost-Sensitive Hierarchical Clustering. 2013.

[23] Serdar Kadioglu, Yuri Malitsky, Meinolf Sellmann, and Kevin Tierney. ISAC –Instance-Specific algorithm configuration. In *Proceedings of the 2010 conference on ECAI 2010: 19th european conference on artificial intelligence*, pages 751–756, NLD, 2010. IOS Press.

[24] Serdar Kadioglu, Yuri Malitsky, Ashish Sabharwal, Horst Samulowitz, and Meinolf Sellmann. Algorithm Selection and Scheduling. In Jimmy Lee, editor, *Principles and Practice of Constraint Programming – CP 2011*, volume 6876, pages 454–469. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011. Series Title: Lecture Notes in Computer Science.

[25] Mark Roberts and Adele E. Howe. Directing a portfolio with learning. In *Proceedings of the workshop on learning for search at the twenty-first national conference on artificial intelligence*, 2006.

[26] Mauro Vallati, Luka Chrpa, and Diane Kitchin. An Automatic Algorithm Selection Approach for Planning. In *2013 IEEE 25th International Conference on Tools with Artificial Intelligence*, pages 1–8, Herndon, VA, November 2013. IEEE.

[27] Alfonso Gerevini, Alessandro Saetti, and Mauro Vallati. Planning through automatic portfolio configuration: The PbP approach. *The Journal of Artificial Intelligence Research (JAIR)*, 50, July 2014.

[28] Hans Degroote, Bernd Bischl, Lars Kotthoff, and Patrick De Causmaecker. Reinforcement learning for automatic online algorithm selection - an empirical study. In *ITAT*, 2016.

[29] Hans Degroote, Patrick De Causmaecker, Bernd Bischl, and Lars Kotthoff. A regression-based methodology for online algorithm selection. In *SOCS*, 2018.

[30] Juan A. Besada, Ivan Campana, Luca Bergesio, Ana M. Bernardos, and Gonzalo de Miguel. Drone Flight Planning for Safe Urban Operations: UTM Requirements and Tools. In *2019 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*, pages 924–930, Kyoto, Japan, March 2019. IEEE.

[31] Steven Michael LaValle. *Planning Algorithms*. Cambridge University Press, Cambridge ; New York, 2006. OCLC: ocm65301992.

[32] John R. Rice. The Algorithm Selection Problem. In *Advances in Computers*, volume 15, pages 65–118. Elsevier, 1976.

[33] S. M. Weiss and N. Indurkhya. Rule-based Machine Learning Methods for Functional Prediction. *Journal of Artificial Intelligence Research*, 3:383–403, December 1995.

[34] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Information Science and Statistics. Springer, New York, 2006.

[35] Akash Manjunath, Manoj Bhat, Klym Shumaiev, Andreas Biesdorf, and Florian Matthes. Decision Making and Cognitive Biases in Designing Software Architectures. In *2018 IEEE International Conference on Software Architecture Companion (ICSA-C)*, pages 52–55, Seattle, WA, April 2018. IEEE.

[36] Rahul Mohanani, Iflaah Salman, Burak Turhan, Pilar Rodriguez, and Paul Ralph. Cognitive Biases in Software Engineering: A Systematic Mapping Study. *IEEE Transactions on Software Engineering*, pages 1–1, 2018.

[37] Tom Schaul and Juergen Schmidhuber. Metalearning. *Scholarpedia*, 5(6):4650, 2010.

[38] Francis Colas, Srivatsa Mahesh, Francois Pomerleau, Ming Liu, and Roland Siegwart. 3D path planning and execution for search and rescue ground robots. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 722–727, Tokyo, November 2013. IEEE.

[39] Jean Berger and Nassirou Lo. An innovative multi-agent search-and-rescue path planning approach. *Computers & Operations Research*, 53:24–31, January 2015.

[40] Karl Obermeyer. Path Planning for a UAV Performing Reconnaissance of Static Ground Targets in Terrain. In *AIAA Guidance, Navigation, and Control Conference*, Chicago, Illinois, August 2009. American Institute of Aeronautics and Astronautics.

[41] Karl J. Obermeyer, Paul Oberlin, and Swaroop Darbha. Sampling-Based Path Planning for a Visual Reconnaissance Unmanned Air Vehicle. *Journal of Guidance, Control, and Dynamics*, 35(2):619–631, March 2012.

[42] John Reif and Micha Sharir. Motion planning in the presence of moving obstacles. In *26th Annual Symposium on Foundations of Computer Science (sfcs 1985)*, pages 144–154, Portland, OR, USA, 1985. IEEE.

[43] Mohammadreza Radmanesh, Manish Kumar, Paul H. Guentert, and Mohammad Sarim. Overview of Path-Planning and Obstacle Avoidance Algorithms for UAVs: A Comparative Study. *Unmanned Systems*, 06(02):95–118, April 2018.

[44] D. Rathbun, S. Kragelund, A. Pongpunwattana, and B. Capozzi. An evolution based path planning algorithm for autonomous motion of a UAV through uncertain environments. In *Proceedings. The 21st Digital Avionics Systems Conference*, volume 2, pages 8D2–1–8D2–12, Irvine, CA, USA, 2002. IEEE.

[45] Sergiy Butenko, Robert Murphey, and Panos M. Pardalos, editors. *Cooperative Control: Models, Applications and Algorithms*, volume 1 of *Cooperative Systems*. Springer US, Boston, MA, 2003.

[46] Navid Dadkhah and Bérénice Mettler. Survey of Motion Planning Literature in the Presence of Uncertainty: Considerations for UAV Guidance. *Journal of Intelligent & Robotic Systems*, 65(1-4):233–246, January 2012.

[47] Niklas Peinecke and Alexander Kuenz. Deconflicting the Urban Drone Airspace. In *2017 IEEE/AIAA 36th Digital Avionics Systems Conference (DASC)*, pages 1–6, St. Petersburg, FL, September 2017. IEEE.

[48] Cosme A. Ochoa and Ella M. Atkins. Fail-Safe Navigation for Autonomous Urban Multicopter Flight. In *AIAA Information Systems-AIAA Infotech @ Aerospace*, Grapevine, Texas, January 2017. American Institute of Aeronautics and Astronautics.

[49] Shyba Zaheer, Jayaraju M, and Tauseef Gulrez. Performance Analysis of Path Planning Techniques for Autonomous Mobile Robots. In *2015 IEEE International Conference on Electrical, Computer and Communication Technologies (ICECCT)*, pages 1–5, Coimbatore, India, March 2015. IEEE.

[50] Mark Moll, Ioan A. Sucan, and Lydia E. Kavraki. Benchmarking Motion Planning Algorithms: An Extensible Infrastructure for Analysis and Visualization. *IEEE Robotics & Automation Magazine*, 22(3):96–102, September 2015.

[51] Mehmet Korkmaz and Akif Durdu. Comparison of Optimal Path Planning Algorithms. In *2018 14th International Conference on Advanced Trends in Radioelecrtronics, Telecommunications and Computer Engineering (TCSET)*, pages 255–258, Lviv-Slavske, Ukraine, February 2018. IEEE.

[52] B.K. Patle, Ganesh Babu L, Anish Pandey, D.R.K. Parhi, and A. Jagadeesh. A Review: On Path Planning Strategies for Navigation of Mobile Robot. *Defence Technology*, 15(4):582–606, August 2019.

[53] Adele E. Howe, Eric Dahlman, Christopher Hansen, Michael Scheetz, and Anneliese von Mayrhauser. Exploiting Competitive Planner Performance. In Gerhard Goos, Juris Hartmanis, Jan van Leeuwen, Susanne Biundo, and Maria Fox, editors, *Recent Advances in AI Planning*, volume 1809, pages 62–72. Springer Berlin Heidelberg, Berlin, Heidelberg, 2000. Series Title: Lecture Notes in Computer Science.

[54] L. E. Dubins. On Curves of Minimal Length with a Constraint on Average Curvature, and with Prescribed Initial and Terminal Positions and Tangents. *American Journal of Mathematics*, 79(3):497, July 1957.

[55] Nils Nilsson. A Mobius Automation: an Application of Artificial Intelligence Techniques. *IJCAI'69: Proceedings of the 1st international joint conference on Artificial intelligence*, pages 509–520, May 1969.

[56] Sertac Karaman and Emilio Frazzoli. Sampling-Based Algorithms for Optimal Motion Planning. *The International Journal of Robotics Research*, 30(7):846–894, June 2011.

[57] Jonathan D. Gammell, Siddhartha S. Srinivasa, and Timothy 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*, pages 2997–3004, Chicago, IL, September 2014. IEEE.

[58] Lisa Fern, Robert C. Rorie, and Robert Shively. UAS Contingency Management: The Effect of Different Procedures on ATC in Civil Airspace Operations. In *14th AIAA Aviation Technology, Integration, and Operations Conference*, Atlanta, GA, June 2014. American Institute of Aeronautics and Astronautics.

[59] 3dr - drone & uav technology - fail-safe overview. https://3dr.com/kb/failsafe-overview/ Accessed May 30, 2016.

[60] Mia N. Stevens and Ella M. Atkins. Multi-Mode Guidance for an Independent Multicopter Geofencing System. In *16th AIAA Aviation Technology, Integration, and Operations Conference*, Washington, D.C., June 2016. American Institute of Aeronautics and Astronautics.

[61] James Luxhoj. System Safety Modeling of Alternative Geofencing Configurations for small UAS. *International Journal of Aviation, Aeronautics, and Aerospace*, 2016.

[62] Open data - nyc.gov, 2016. http://www1.nyc.gov/site/planning/data-maps/open-data.page Accessed Dec 1, 2016.

[63] Yoav Gabriely and Elon Rimon. Spanning-tree based coverage of continuous areas by a mobile robot. *Annals of Mathematics and Artificial Intelligence*, 31(1/4):77–98, 2001.

[64] R. C. Prim. Shortest Connection Networks And Some Generalizations. *Bell System Technical Journal*, 36(6):1389–1401, November 1957.

[65] Mark Roberts, Adele Howe, and On Flom. Learned models of performance for many planners. In *In ICAPS 2007, workshop AI planning and learning*, 2007.

[66] Mark Roberts and Adele Howe. Learning from planner performance. *Artificial Intelligence*, 173(5-6):536–561, April 2009.

[67] Abhinav Saxena, Jose Celaya, Bhaskar Saha, Sankalita Saha, and Kai Goebel. Evaluating algorithm performance metrics tailored for prognostics. In *2009 IEEE Aerospace conference*, pages 1–13, Big Sky, MT, USA, March 2009. IEEE.

[68] A. Jain and D. Zongker. Feature selection: evaluation, application, and small sample performance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(2):153–158, February 1997.

[69] Stuart J. Russell, Peter Norvig, and Ernest Davis. *Artificial Intelligence: A Modern Approach*. Prentice Hall Series in Artificial Intelligence. Prentice Hall, Upper Saddle River, 3rd ed edition, 2010.

[70] Tejas Puranik, Evan Harrison, Imon Chakraborty, and Dimitri Mavris. Aircraft Performance Model Calibration and Validation for General Aviation Safety Analysis. *Journal of Aircraft*, pages 1–11, March 2020.

[71] N. H. McClamroch. *Steady aircraft flight and performance*. Princeton University Press, Princeton, N.J, 2011. OCLC: ocn611551579.

[72] Sweewarman Balachandran and Ella M. Atkins. Flight Safety Assessment and Management to Prevent Loss of Control Due to In-Flight Icing. In *AIAA Guidance, Navigation, and Control Conference*, San Diego, California, USA, January 2016. American Institute of Aeronautics and Astronautics.

[73] Afshin Mardani, Marcello Chiaberge, and Paolo Giaccone. Communication-Aware UAV Path Planning. *IEEE Access*, 7:52609–52621, 2019.

[74] Shaunak D. Bopardikar, Brendan Englot, and Alberto Speranzon. Multiobjective path planning in GPS denied environments under localization constraints. In *2014 American Control Conference*, pages 1872–1879, Portland, OR, USA, June 2014. IEEE.

[75] I.M. Mitchell and S. Sastry. Continuous path planning with multiple constraints. In *42nd IEEE International Conference on Decision and Control (IEEE Cat. No.03CH37475)*, pages 5502–5507, Maui, HI, USA, 2003. IEEE.

[76] Shashi Mittal and Kalyanmoy Deb. Three-dimensional offline path planning for UAVs using multiobjective evolutionary algorithms. In *2007 IEEE Congress on Evolutionary Computation*, pages 3195–3202, Singapore, September 2007. IEEE.

[77] Alexis Guigue, Mojtaba Ahmadi, Rob Langlois, and M. John Hayes. Pareto Optimality and Multiobjective Trajectory Planning for a 7-DOF Redundant Manipulator. *IEEE Transactions on Robotics*, 26(6):1094–1099, December 2010.

[78] M. Haklay and P. Weber. OpenStreetMap: User-Generated Street Maps. *IEEE Pervasive Computing*, 7(4):12–18, October 2008.

[79] Thomas Sean Kelso. CelesTrak, 1985.

[80] Brandon Rhodes. Skyfield: Generate high precision research-grade positions for stars, planets, moons, and Earth satellites, 2020.

[81] US Census Bureau . TIGER 2010 Census Block State-based Shapefile with Housing and Population Data, 2010.

[82] Per K. Enge. The Global Positioning System: Signals, measurements, and performance. *International Journal of Wireless Information Networks*, 1(2):83–105, April 1994.

[83] Rock Santerre. Impact of GPS satellite sky distribution. *Manuscripta Geodaetica*, 16, January 1991.

[84] Richard B Langley and others. Dilution of precision. *GPS world*, 10(5):52–59, 1999.

[85] Justin R. Rufa and Ella M. Atkins. Unmanned Aircraft System Navigation in the Urban Environment: A Systems Analysis. *Journal of Aerospace Information Systems*, 13(4):143–160, April 2016.

[86] Justin Fung. Manhattan Population Explorer, May 2019.

[87] NYC Department of City Planning. Community Districts, January 2013.

[88] Jean-Claude Latombe. *Robot Motion Planning*. 1991. OCLC: 1053844989.

[89] David Gonzalez, Joshue Perez, Vicente Milanes, and Fawzi Nashashibi. A Review of Motion Planning Techniques for Automated Vehicles. *IEEE Transactions on Intelligent Transportation Systems*, 17(4):1135–1145, April 2016.

[90] Brian Paden, Michal Cap, Sze Zheng Yong, Dmitry Yershov, and Emilio Frazzoli. A Survey of Motion Planning and Control Techniques for Self-Driving Urban Vehicles. *IEEE Transactions on Intelligent Vehicles*, 1(1):33–55, March 2016.

[91] Tomás Lozano-Pérez and Michael A. Wesley. An algorithm for planning collision-free paths among polyhedral obstacles. *Communications of the ACM*, 22(10):560–570, October 1979.

[92] James Reeds and Lawrence Shepp. Optimal Paths for a Car that goes both Forwards and Backwards. *Pacific Journal of Mathematics*, 145(2):367–393, October 1990.

[93] Peter Hart, Nils Nilsson, and Bertram Raphael. A Formal Basis for the Heuristic Determination of Minimum Cost Paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2):100–107, 1968.

[94] E. W. Dijkstra. A Note on Two Problems in Connexion with Graphs. *Numerische Mathematik*, 1(1):269–271, December 1959.

[95] Sven Koenig, Maxim Likhachev, and David Furcy. Lifelong Planning A*. *Artificial Intelligence*, 155(1-2):93–146, May 2004.

[96] Maxim Likhachev, Geoff Gordon, and Sebastian Thrun. ARA*: Formal analysis. Technical report, 2003.

[97] S. Koenig and M. Likhachev. Fast Replanning for Navigation in Unknown Terrain. *IEEE Transactions on Robotics*, 21(3):354–363, June 2005.

[98] Dave Ferguson and Anthony Stentz. Field D*: An Interpolation-Based Path Planner and Replanner. In Sebastian Thrun, Rodney Brooks, and Hugh Durrant-Whyte, editors, *Robotics Research*, volume 28, pages 239–253. Springer Berlin Heidelberg, Berlin, Heidelberg, 2007.

[99] K. Daniel, A. Nash, S. Koenig, and A. Felner. Theta*: Any-Angle Path Planning on Grids. *Journal of Artificial Intelligence Research*, 39:533–579, October 2010.

[100] L.E. Kavraki, P. Svestka, J.-C. Latombe, and M.H. Overmars. Probabilistic Roadmaps for Path Planning in High-Dimensional Configuration Spaces. *IEEE Transactions on Robotics and Automation*, 12(4):566–580, August 1996.

[101] Steven M. LaValle. Rapidly-Exploring Random Trees: A New Tool for Path Planning. 1998.

[102] J.J. Kuffner and S.M. LaValle. RRT-connect: An efficient approach to single-query path planning. In *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065)*, volume 2, pages 995–1001, San Francisco, CA, USA, 2000. IEEE.

[103] Lucas Janson, Edward Schmerling, Ashley Clark, and Marco Pavone. Fast marching tree: A fast marching sampling-based method for optimal motion planning in many dimensions. *The International Journal of Robotics Research*, 34(7):883–921, June 2015.

[104] I. T. Kiguradze. Boundary-value problems for systems of ordinary differential equations. *Journal of Soviet Mathematics*, 43(2):2259–2339, October 1988.

[105] K. Spindler. Motion planning via optimal control theory. In *Proceedings of the 2002 American Control Conference (IEEE Cat. No.CH37301)*, pages 1972–1977 vol.3, Anchorage, AK, USA, 2002. IEEE.

[106] Yang Wang and Stephen Boyd. Fast Model Predictive Control Using Online Optimization. *IEEE Transactions on Control Systems Technology*, 18(2):267–278, March 2010.

[107] T. Howard, M. Pivtoraiko, R. A. Knepper, and A. Kelly. Model-predictive motion planning: Several key developments for autonomous mobile robots. *IEEE Robotics Automation Magazine*, 21(1):64–73, 2014.

[108] Chang Liu, Seungho Lee, Scott Varnhagen, and H. Eric Tseng. Path Planning for Autonomous Vehicles using Model Predictive Control. In *2017 IEEE Intelligent Vehicles Symposium (IV)*, pages 174–179, Los Angeles, CA, USA, June 2017. IEEE.

[109] Pascal Kerschke, Holger H. Hoos, Frank Neumann, and Heike Trautmann. Automated Algorithm Selection: Survey and Perspectives. *Evolutionary Computation*, 27(1):3–45, March 2019.

[110] Mario A. Muñoz, Michael Kirley, and Saman K. Halgamuge. The Algorithm Selection Problem on the Continuous Optimization Domain. In Christian Moewes and Andreas Nürnberger, editors, *Computational Intelligence in Intelligent Data Analysis*, volume 445, pages 75–89. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013. Series Title: Studies in Computational Intelligence.

[111] K. Pearson. "Das Fehlergesetz und Seine Verallgemeinerungen Durch Fechner und Pearson." A Rejoinder. *Biometrika*, 4(1-2):169–212, June 1905.

[112] Marlin P. Strub and Jonathan D. Gammell. Adaptively Informed Trees (AIT*): Fast Asymptotically Optimal Path Planning through Adaptive Heuristics. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3191–3198, Paris, France, May 2020. IEEE.

[113] V.D. Gligor and S.H. Shattuck. On Deadlock Detection in Distributed Systems. *IEEE Transactions on Software Engineering*, SE-6(5):435–440, September 1980.

[114] M. Singhal. Deadlock detection in distributed systems. *Computer*, 22(11):37–48, November 1989.

[115] P. Dadvar and K. Skadron. Potential thermal security risks. In *Semiconductor Thermal Measurement and Management IEEE Twenty First Annual IEEE Symposium, 2005.*, pages 229–234, San Jose, CA, USA, 2005. IEEE.

[116] D. Mohla, L.B. McClung, and N.R. Rafferty. Electrical safety by design. In *Industry Applications Society 46th Annual Petroleum and Chemical Technical Conference (Cat.No. 99CH37000)*, pages 363–369, San Diego, CA, USA, 1999. IEEE.

[117] B.W. Boehm. Software risk management: principles and practices. *IEEE Software*, 8(1):32–41, January 1991.

[118] A.H. Bonnett. Root cause AC motor failure analysis with a focus on shaft failures. *IEEE Transactions on Industry Applications*, 36(5):1435–1448, October 2000.

[119] F. Richardeau, P. Baudesson, and T.A. Meynard. Failures-tolerance and remedial strategies of a PWM multicell inverter. *IEEE Transactions on Power Electronics*, 17(6):905–912, November 2002.

[120] Julio Cesar Graves, Wallace Hessler Leal Turcio, Jorge Alvarez, and Takashi Yoneyama. Spectral Signatures of Pneumatic Actuator Failures: Closed-Loop Approach. *IEEE/ASME Transactions on Mechatronics*, 23(5):2218–2228, October 2018.

[121] Joris Jaguemont, Loic Boulon, Yves Dube, and Francois Martel. Thermal Management of a Hybrid Electric Vehicle in Cold Weather. *IEEE Transactions on Energy Conversion*, 31(3):1110–1120, September 2016.

[122] Simon Watkins, Abdulghani Mohamed, and Michael V. Ol. Gusts Encountered by MAVs in Close Proximity to Buildings. In *AIAA Scitech 2019 Forum*, San Diego, California, January 2019. American Institute of Aeronautics and Astronautics.