Development of Risk-Informed UAV Pathfinder Based on A* Algorithm

Micko Lesmana

11201901010

Undergraduated Thesis

Department of Aviation Engineering
International University Liaison Indonesia
BSD City
7 July 2024

Approval Page

Report

Risk Informed UAV Pathfinding, UPX

3 December 2022

Micko Lesmana

11201901010

Department of Aviation Engineering

Acknowledge by:

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS

A*	A-star
AIS	Abbreviated Injury Scale
API	Application Programming Interface
BBN	Bayesian Belief Networks
BC	Blunt Criteria
CFIT	Controlled Flight Into Terrain
CRS	Coordinate Reference System
DOJC	Dropped or Jettisoned Component
DoD	United States Department of Defense
ETA	Event Tree Analysis
FAA	Federal Aviation Administration
FMECA	Failure Mode, Effects and Criticality Analysis
FTA	Fault Tree Analysis
GHSL	Global Human Settlement Layer
GIS	Geographic Information System
GPS	Global Positioning System
GRM	Ground Risk Model
geoJSON	Geographic JavaScript Object Notation
ICAO	International Civil Aviation Organization
KE	Kinetic Energy
LOC	Lost of Control
MTBF	Mean Time Between Failure
OSM	OpenStreetMap
PDF	Probability Density Function
RRT	Rapidly Exploring Random Tree
UDS	Unpremediated Descent Scenario

UAS	Unmanned Aerial System
UAV	Unmanned Aerial Vehicle
UPX	UAV Pathfinding eXtension
CE	Casualty Expectation
AGL	Above Ground Level
BFS	Bread-First Search
DFS	Depth-First Search
ACO	Ant Colony Optimization
AI	Artificial Intelligence
GBFS	Greedy Best-First Search

CHAPTER 1
INTRODUCTION

1.1 Introduction

Unmanned Aerial Vehicle (UAV) also known as a drone, is getting more recognition in recent years. The booming of UAV is caused by the component needed to manufacture the UAV, especially microcontrollers getting cheaper, and smaller Austin and Brewster, 2024. Compare to previous UAV iteration, UAV nowadays offers much wider capability, such as enhanced range, the ability to mount hardware, and camera for example, and of course easily deployable without requiring many technicians to maintain them.

UAV are often being used in area where it is dangerous, impractical, and expensive for human. The affordable entry point of using a UAV incentivized emerging industries to exploit the capability that UAV offer. Not only that but the existing industry is also getting interested to use UAV as an augmentation to its existing operational infrastructure. And that industries are, logistics companies, agricultural companies, multimedia companies, and surveyors to name a few DJI, 2023.

Although it provides a great solution, UAV have the potential to pose risks to people, property, and other aircraft. Many countries have regulations in place that require UAV to be operated in a safe and responsible manner FAA, 2016. Risk-based UAV path planner can help to ensure that UAV are operated safely and responsibly. By considering potential risks and hazards, a risk-based approach can help to reduce the likelihood of accidents or incidents occurring during the flight. By considering potential risks and hazards, this approach can help to reduce the likelihood of accidents or incidents occurring during deployment.

Other than its impacts towards public, UAV can also potentially impact the environment, for example by disturbing wildlife or damaging natural habitats Holland, 2015. UAV path planner can help to minimize these impacts by establishing rules around the use of UAV in sensitive areas.

1.2 Research purpose

For ensuring the safety and efficiency of UAV operations, this research aims to develop a UAV trajectory planning based on the risk assessment on the ground.

1.3 Research scope

- UAV model that being used in this research are fixed wing and multirotor.
- This research only use point mass representation with generalize aerodynamic equation.

- Based on the point above, UAV failure is assumed to be only power failure, without damage to its shape.
- Risk assessed in this paper are combination of human exposed area and area that can cause major accident such as airfields, and power station.
- This research use Jakarta, Indonesia as its geographic location, since it has varying building density, rivers, and randoml patch of lands.

1.4 Research problem

- There are risks associated when of UAV falls on the ground.
- Modeling physical interaction of UAV crash.
- Selection of pathfinding algorithm to get optimum time.

1.5 Significance of the study

UAV Pathfinder application could be one of the toolkit for UAV operator to plan their safe operation of flying their UAV, reducing inccident and accident. Developer and researcher could use the library itself as an extension or starting point in the development of the Unmanned Aerial System (UAS) Traffic Management application.

1.6 Research question

- Which kind of geographic data and data processing model perfectly suited for pathfinding risk assesment.
- How to statistically create the best approximation of crash dynamics.
- What kind of pathfinding method that produces the quasi-optimal path without sacrificing computational time.

1.7 Thesis Structure

CHAPTER 2
LITERATURE REVIEW

2.1 Overview

In this part of the literature review, we will discuss two main concepts: the Ground Risk Model and the Path Finder. in the first part , we will examine the Ground Risk Model, which involves discussing the concept of probability within a spatial context. Consider this scenario: if we want to calculate the likelihood of someone being hit by a drone in a specific location, how do we model this phenomenon? As you can see, scenarios like this are key to understanding the Ground Risk Model.

Next, we will explore the Path Finder. After identifying a model that answers the question of where a UAV is most likely to hit someone, we move on to the next question: how do we navigate the UAV or provide the safest route to minimize the risk of severe casualties? This is why we will discuss the Risk-Based Path Finder.

2.2 Ground risk model

A ground risk model is a type of assessment tool that is used to evaluate the potential risks associated with the operation of UAV. This model takes into account a variety of factors, including the physical environment in which the UAV will be operating, the capabilities, and the limitations of the UAV itself. The goal of a ground risk model is to identify and prioritize potential risks in order to produce inform decision-making and risk management strategies for UAV operations.

We found a total of 22 different research papers that contributed to our study on ground risk models. Various approaches have been taken to develop ground risk models for UAVs. One approach is the use of scenario-based methods, as discussed in the research of Ancel et al., 2017, which involves the development of detailed scenarios that describe potential UAV operations and the associated risks.

To clarify further, in the research done by Ancel et al., 2017, there are two common methods for describing the severity of accidents using scenario-based approaches. These scenarios are typically coupled with safety and hazard levels on a per-operation basis. The first method utilizes a lookup table to determine the severity of an accident. The second method represents the scenario as a directed acyclic graph, a type of tree graph, where we start at the root node and traverse based on yes or no questions. These questions can range from simple ones, such as whether the UAV hit a person, to more detailed ones, such as whether the accident caused harm.

This method is computed using techniques such as risk tree diagrams, Bayesian belief networks Ancel et al., 2017, and hazard risk assessment matrices Barr et al., 2017. Although these methods might produce different analyses, they generally

describe the same phenomena.

From the description above, it appears that scenario-based models do not suit our needs for a spatially based ground risk model. Essentially, they reduce to questions of general progression from the point of impact to the accident caused by the drone. They provide no information on how the position of the UAV affects the casualty risk of failed operations and only serve as a preliminary assessment.

To add a spatial component to our models, we can simply incorporate location information into the yes/no question tree graph. What I mean by this is to propose an idea where, in a certain area, if a UAV somehow falls, does it cause harm, and based on the surrounding environment, does that harm result in fatality?

This brings us to the most widely used method, so called ground probabilistic risk assessment methods, which involve the quantification of risks per area using probability and consequence metrics. Although a greater number of research papers use this method, as compiled by Washington et al., 2017, our concept shown in figure 2.1 is derived from the research conducted by Primatesta, Rizzo, and la Cour-Harbo, 2020.

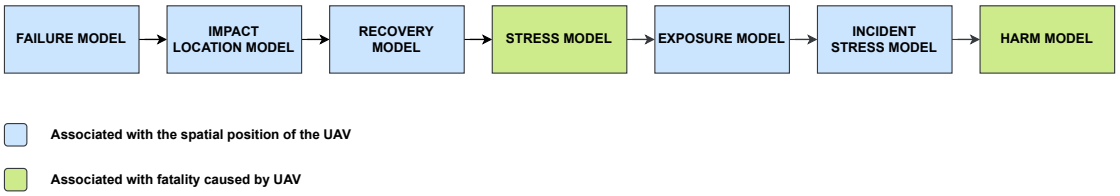


Figure 2.1: Components of ground risk model adopted from Primatesta, Rizzo, and la Cour-Harbo, 2020

A ground probabilistic risk assessment model, shortened to ground risk model, of a UAV typically consists of several subcomponents that are used to assess various factors that can impact the safety and effectiveness of UAV operations. These components in the ground risk model include the failure model, impact location model, recovery model, stress model, exposure model, incident stress model, and harm model, which will be explained further below.

However, since this is the most widely used approach, it is not surprising that this approach is not new, as many earlier researchers have already followed guidelines to model these components. For example, Breunig et al., 2018 provides a general model guideline. These guidelines focus more or less on the probability of said accidents causing fatalities but do not compute the overall area risk.

Washington et al., 2017 offers a meta-analysis of different kinds of research using more or less the same paradigm to describe the ground risk model. Washington et al., 2017 itself uses the method of its analysis based on the guidelines from Pat-Cornell, 1996, which studied the level of analytical detail when modeling risk. Pat-Cornell, 1996 provides us with a degree of certainty ranging from one to six, where one uses

a very simple model and six accounts for every possible situation that can affect the model.

After a thorough discussion, we derived our study based on Primatesta, Rizzo, and la Cour-Harbo, 2020 simply because it shows the most comprehensive way to calculate the ground risk model, which can be summarized into Equation 2.1.

$$P_{fatal}(x) = P_{failure}(x) \cdot P_{impact}(x) \cdot P_{recovery} \cdot P_{stress} \cdot P_{exposure}(x) \cdot P_{incident}(x) \cdot P_{harm}(x) \tag{2.1}$$

All of these models are represented as the probability of the said event-model occurring. Where P is a probability of an event and x is the location on a grid map. If we take a closer look at Equation 2.1, some P_n terms do not contain any location parameter. This is because those probabilities are applied as global variables to every area.

Back to the model, since all models occur consecutively from the failure model to the harm model, each component of the ground risk model is multiplied by the others. Although many papers do not use all of these components in their ground risk models and often use different methodologies to analyze the same component, the fundamental approach remains consistent. For example, Primatesta, Rizzo, and la Cour-Harbo, 2020 only use the event, impact location, and fatality models, while Kim and Bae, 2022 use the event, impact, exposure, incident stress, and harm models. Despite apparent differences in modeling ground risk, researchers sometimes combine multiple subcomponents into one. For instance, in Primatesta, Rizzo, and la Cour-Harbo, 2020, the fatality model component is, in fact, a combination of the incident, stress, and harm models. All the models in ground risk assessment are explained in following sub-chapters.

2.2.1 Failure model

A failure model is the first subcomponent in the ground risk model that evaluates the potential for failures in UAV. This potentially leads to a crash impact in a given variety of factors that can contribute to UAV failures, including design and manufacturing defects, operational errors, and environmental factors.

The failure model can help to understand on how the UAV would behave after experiencing the failure, Washington et al., 2017 described four distinct events following the failure. Which are Unpremediated Descent Scenario (UDS), Lost of Control (LOC), Controlled Flight Into Terrain (CFIT), and Dropped or Jettisoned Component (DOJC).

All of these failure events describe how the UAV crashes into the ground. UDS is a mode where the drone automatically enters a safe mode and tries to land as safely as possible. Drone manufacturers often implement this feature when critical components are detected to be defective. UDS can also be triggered when the UAV

loses its signal to its pilot. Most mid- and high-priced drones usually have this feature, such as those from DJI, 2024.

Controlled flight into terrain occurs when the UAV still has flight authority, but its main propulsion is no longer working or its battery is at very low power, allowing it to only move its control surface servos. It is similar to UDS, except that in UDS, the UAV still has propulsion power. Also like UDS, if the UAV is advanced enough Nemire, 2015, it can auto-navigate to a safer area.

Loss of control scenario occurs when the UAV is no longer responsive and has lost its navigation system, causing it to fly away while maintaining its velocity and altitude until it runs out of energy or crashes into terrain. This condition is much more degraded compared to UDS. Lastly, the "dropped or jettisoned" scenario, probably the least common failure stated by Washington et al., 2017, occurs when some components detach from the drone and fall to the ground.

Although the definition of a ways that UAV can fell into the terrain, it does not necessarily help us calculate the likelihoodness of a UAV to fell into the ground. With that being said this definition can help us to formulate the descent model of the ground risk model.

If we want to calculate the probability of the drone starts to fail, there are two main models that we can use. The first one is failure model proposed by Breunig et al., 2018 and Weibel and Hansman, 2012 where it is based on historical data. Data driven historical data are using previous UAV crash, and its UAV specification as a starting point for data interpolation and extrapolation for calculating so called Mean Time Between Failure (MTBF).

Unfortunately since UAV industries is in its infancy and an added fact with many variants of UAV already exist in short period of time. There are not enough historical data to make an interpolation, which is an issue commented by Breunig et al., 2018, many model using an MTBF simply use a standard assumed by ARC, 2015 which is average of 1 failure per 100 hours of flight for small UAV.

Although with its limitation, there are few research that trying to emulate MTBF to solve target level of safety to help certifying authorities to make a propper judgment in regards to ever growing small UAV industry. Such model is developed by Burke et al., 2011, where they make a system-level airworthiness tools that predict how safe a UAV is based on the sizes of the aircraft which is explained the equation 2.2 below.

$$MTBF = \frac{\rho\pi b^2}{2.78} = k(\rho\pi b^2) \tag{2.2}$$

Where ρ is population density, b is geometric wingspan of the drone, and k is population correction factor, where k is $\frac{1}{2.78}$ in a setting where the population density is calculated per square miles. This equation is derived from target level

safety algorithm proposed by Burke et al., 2011. Which then derived from Casualty Expectation (CE) equation 2.3

$$CE = P_f \times P_d \times A_l \times P_k \times S \tag{2.3}$$

Where CE is expected casualty rate per flight hours, P_f is the probability of failure in failure per flight hour, P_d is the population density in people per square mile, A_l is the dimensionless lethal area, P_k is fatality probability, and S is the dimensionless shelter factor. We then change the P_d as a ρ to signified the average in time for population in a given mission area. A_l is modeled from using the maximum circular area of the UAV itself which is the maximum wing span b in feet or in the case of multirotor, maximum diagonal distance. Therefore A_l can be substitute into πb^2. Pointer given by Burke et al., 2011 simply using A_l, which calculate in feet, will give an error when being multiplied by ρ where it is in square miles. Therefore Burke et al., 2011 give an error correction to divide ρ with the number of 1 ft^2 in a 1 $mile^2$ which roughly is 27.8 million.

Burke et al., 2011 normalized the P_k to 1 to argue based on the fact provided by ARC, 2015. This normalization came from the calculation of kinetic energy for small UAV with a weight of 64 lb with the speed of 100 ft/s, in which the UAV is still in non-lethal configuration. Move on to the S, it describe to be in range where 1 is non-covered area and 0 is fully sheltered. Fully sheltered in this case is when an UAV with the weight of 64 lb and the speed of 100 ft/s crash into a populated building, the occupants will be in no harm condition. If we rearrange the equation and set the CE into 1×10^{-7} which also has the same meaning with equivalent level safety, where we will get resulting equation 2.4

$$10^{-7} = P_f \times \frac{\rho}{2.78 \times 10^{-7}} \times \pi b^2 \tag{2.4}$$

Since P_f is probabilty of failure per flight hours which also share the same meaning with MTBF, we can change the P_f into MTBF. And if we rearrange the equation one more time we will get the same equation of 2.2. However since that equation is in US customary unit, we must convert it into metric system, where we divided 1 m^2 per 1 km^2. Fortunately it is quite simple, as we just need to change error correction factor from 2.78×10^{-7} into 10^{-6} which resulted into equation 2.5

$$MTBF = \frac{\rho \pi b^2}{10} = k(\rho \pi b^2) \tag{2.5}$$

Aside from system-level airworthiness tool, there are systematic methodologies used in risk management and safety engineering to assess the failure of UAV. These tools are Fault Tree Analysis (FTA), Event Tree Analysis (ETA) , and Failure Mode, Effects and Criticality Analysis (FMECA) Barr et al., 2017. Where focuses on identifying root causes of specific failures using logical diagrams. ETA evaluates the

outcomes of initiating events and their subsequent paths. And FMECEA identifies potential failure modes, their effects, and prioritizes them based on criticality.

Also there is a new development using a machine learning model such as Bayesian Belief Networks (BBN) Ancel et al., 2017 to be used for failure analysis. Where Ancel et al., 2017 provide the most detailed failure risk model incorporating real-time monitoring of UAV component status to determine the likelihoodness of failure. BBN itself is a graphical model used for probabilistic reasoning and decision-making, which represents relationships among variables through directed acyclic graphs. In the context of UAV failure analysis, BBNs can help identify the likelihood of various failure modes by modeling the dependencies and interactions between different system components and environmental factors.

The model such as FTA, ETA, and FMECA sadly only giving us an answer to a question of what is and are the cause of the crash. BBN proposed by Ancel et al., 2017 also is a only a framework for implementing realtime decision making when it is in operation. Some researcher also assume a constant MTBF of 100 hours proposed by the ARC, 2015. We also make use the assumption that the failure's probability of happening is constant. Therefore, variables like phase, length, environment circumstances, or mission profile that might affect the chance of failure are not taken into account.

2.2.2 Impact location model

The impact location model describes the position and size of the impact region caused by a UAV crash. Various models are employed to determine this impact area, ranging from single-point impact models to more complex models that account for multiple potential impact points along a UAV's trajectory. The latter strategy establishes a perimeter around which all of the UAV's possible impact points are located, calculated based on the maximum descent trajectory distance. The impact location model can be formulated using empirical or hypothetical approaches.

Empirical models leverage historical aircraft event data as a starting point, using interpolation and extrapolation algorithms to identify potential impact sites. For example, Melnyk et al., 2014 categorizes these models based on the study by Ale and Piers, 2000, further refining impact predictions through weight-based classification.

Where generally speaking, the hypothetical model in research often attempts to establish a relationship between the size of the UAV and the extent of its impacted area. For instance, models proposed by Primatesta, Rizzo, and la Cour-Harbo, 2020 and Burke et al., 2011 utilize aerodynamic principles to simulate impact behavior. This model can be further categorized into ballistic descent, gliding descent, and planform models. The ballistic descent calculates the trajectory based on a ballistic profile, while the gliding model assumes the UAV will continue descending along its glide path. The planform model, on the other hand, uses the aircraft's wing and layout aerodynamic properties to determine the descent trajectory.

According to Primatesta, Scanavino, et al., 2020, it is very costly to calculate full aerodynamic properties per cell in a large grid, which is why many researchers rely on gliding and ballistic descent calculations for impact modeling. Additionally, Primatesta, Rizzo, and la Cour-Harbo, 2020 introduced scenarios such as parachute descent, where the UAV deploys a parachute during descent, and fly-away descent, where the UAV maintains a stable cruise after the pilot loses control.

Primatesta uses a method from the US Department of Transportation FAA, 2000 for calculating the casualty area of a vertically falling inert piece of debris. According to this method, the casualty area is a circle whose radius is the sum of the radius of a circle with an area equal to the largest cross-sectional area of the debris and the radius of a human being (30 cm). For these calculations, an acceptable dimension for a human is considered to be a 182.5 cm tall cylinder with a 30 cm radius. To account for any horizontal velocity component, such as wind or trajectory angles, the basic casualty area can be calculated using the following equation and figures:

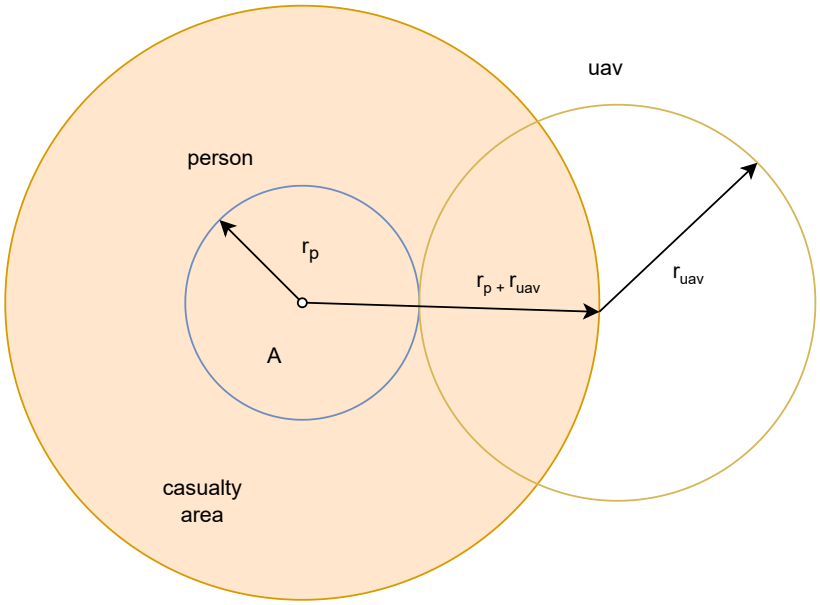


Figure 2.2: Area exposed of casualty area

$$A_{exp}(\theta) = \pi(r_p + r_{uav})^2 \sin(\theta) + (r_p + r_{uav})(h_p + r_{uav}) \cos(\theta) \tag{2.6}$$

Where A_{exp} is an area exposed in m^2, r_p and h_p is the average radius (0.3 m) and average height (182.5 m) of a person respectively in meter, r_{uav} is the radius of the UAV in meter, and θ is the impact angle on the ground. This conservative computation takes into account the maximum radius of the UAV, ensuring a thorough evaluation of the impact area.

2.2.3 Recovery model

A recovery risk model for UAV is a model used to assess the likelihood of successful recovery of a UAV to a optimal or degraded operational state following a failure or malfunction. The model takes into account a variety of factors, the nature and severity of the failure, and the environment in which the recovery is taking place. The recovery risk model also takes into account the UAV design, capabilities and limitations. For example some UAV has parachute failsafe built into it. This parachute is then deployed when the control logic of the UAV detected major system breakdown. Some high-end UAV will enter a automatic landing or controlled crash into terrain before it deploys its parachute. Although not always it is a general rule of thumb that the UAV that has already equiped by parachute usually already has an advance flight controller. Please note that at the time of this writing, there are little to none database containing every specification of each highend drone.

Although there is a significant in studying UAV recovery behaviour, only three researches that managed to include recovery model in its ground risk model. Shelley, 2016 calculated the safest highest possible flight altitude of the UAV based on its weight, or to be precised its resulted kinetic impact. It is showed that under any circumstances UAV with the weight over 1.5 kg must not be flown over a dense population. The study also give a benefit to equiping UAV with parachute will enable most UAV from only permitted to flown in tens of feet to 400 ft Above Ground Level (AGL).

The reasoning behind the small numbers of parachute descent being incorporate into GRM is because it assumes if the UAV already deployed its parachute, there are little to no threat to population. Hence, most of the reasearcher just give if else statements, where to just ignore its calculation if parachute is installed into the UAV. However some researcher goes into detail to see the effect of UAV impacting the surrounding area even with parachute, after all parachute can be drag by wind and still conatains an energy.

Primatesta, Rizzo, and la Cour-Harbo, 2020 use parachute descent calculation based on the study from Bleier et al., 2015, where it calculate possible area where the UAV will land. Bleier et al., 2015 use its calculation based on the relation ship of $s = V \cdot t$, where s is distance, V is velocity and t is time. The relationship is expanded into include wind effect. Which resulted into equation 2.7 and 2.8.

$$t_d = \frac{h}{V_{sink}} \tag{2.7}$$

$$\begin{pmatrix} x \\ y \end{pmatrix} = \left(V_{\text{wind}} \pm \Delta V_{\text{wind}}\right) \frac{h}{V_{\text{sink}}} \cdot \begin{pmatrix} \cos\left(\alpha_{\text{wind}} \pm \Delta\alpha_{\text{wind}}\right) \\ \sin\left(\alpha_{\text{wind}} \pm \Delta\alpha_{\text{wind}}\right) \end{pmatrix} \tag{2.8}$$

Where t_d is time from deployed state to the ground, v_{sink} is the descent velocity,

V_{wind} and ΔV_{wind} is wind speed and random change in wind speed respectively, α_{wind} and $\Delta\alpha_{wind}$ is wind angle and random change in wind angle respectively

We can see how the equation works from figure 2.3 below. Please not that Bleier et al., 2015 only shows the equation for north down east coordinate system, where we use general cartesian coordinate for the figure. Back to the figure, if we view how the UAV is fell from the sky from a top down perspective, where the $0, 0$ is the point of deployment and has a wind affecting its descent trajectory. The UAV will take its time to be dragged into the ground by the wind to the red area, where the left most side is the furthest possible point, and vice versa for the right most side of the red area.

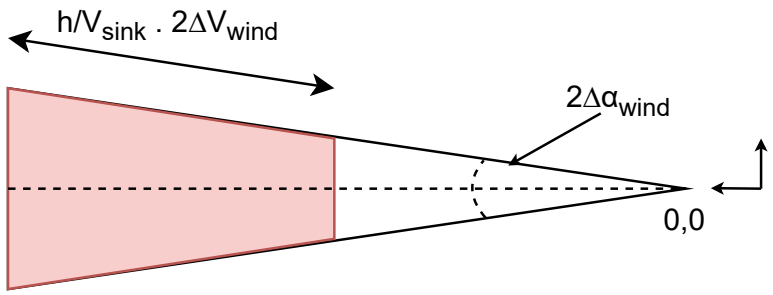


Figure 2.3: Range covered by parachute landing

2.2.4 Stress model

The stress model explains the unpredictability of harmful circumstances occurring at a specific time and place. Or in other term what kind of harm are recieved by the population. For instance, stresses may be described as kinetic energy if the outcome to be calculated was human physical harm modeled in Kinetic Energy (KE). While secondary effects such as explosions and the release of hazardous chemicals will have distinct related stress qualities that need to be modeled differently. There are model proposed by Ball et al., 2012 to analyze the consequences of heat radiation and explosions which often produces by chemical reaction. 18 of 22 of the papers are using only kinetic energy as an only harm type in their ground risk model. This is not surprising since majority of the damage done into human body is kinetic energy based on the bomb explosion effect into human body reserach by Harwick et al., 2007.

Depending on the kind of UAV, different stress factors that relate to one or more harmful processes may exist. Additionally, the UAV's characteristics and its trajectory may be changed to alter the stress profile at the site of impact. Through one or more harm mechanisms, the various stresses may be linked to the undesirable consequence. Examples of physical injury to humans include blunt force, penetration, crushing, blast, burns, lacerations.

It should be mentioned that trauma generated by a blunt force collision is the main cause of injury taken into account by current models. It is possible that blunt

force trauma is not the main cause of injury for UAV with tiny mass. Smaller multirotor UAV, for instance, have a higher tendency to lacerate individuals and inflict physical harm. In the stress model, laceration injuries must be taken into account, according to the Small, 2009.

The KE at impact was calculated by the models using a variety of mass and speed values, which is why the results of the stress models can differ from one another. For instance, Ancel et al., 2017 and other models simply assume the maximum velocity for estimating the impact energy for small UAV flying at low altitudes. Although using terminal velocity would be an overly conservative estimate, Dalamagkidis et al., 2008 evaluation of the impact KE employs a similar assumption.

2.2.5 Exposure model

An exposure model is a method for determining how affected individual or property may be to dangers related to the use of UAV.

A uniform exposure model is the model of population exposure that is most frequently utilized. The use of this approach overlooks possible population clustering within a specified geographic region and, as a result, averages the potential exposure factors contribution to the overall risk assessment. Such methodologies tend to overestimate or underestimate the risk value associated with the current setup because they do not provide sufficient information about the spatial and temporal distribution of the population.

Models such as those created by Burke et al., 2011 and Melnyk et al., 2014 aim to add temporal features, acknowledging how individuals spend their days, taking such factors into consideration. An illustration of a thorough exposure model that emphasizes the time dependencies in population exposure is given by Melnyk et al., 2014. Research conducted by Klepeis et al., 2001 found that individuals spent 68.7% of their time at home, 7.6% outside, 5.5% in cars, 5.4% at factories or offices, and another 12.8% in various indoor settings. This data is used by Melnyk et al., 2014 to specify their exposure model along with the information.

As expected population density data gathered for exposure model came from census, geographical data from satellite imagery. Depending on when each study was done and what regions were being analyzed, a different census data source was used. For instance, Burke et al., 2011 used data from the U.S. Census Bureau, and Clothier et al., 2007 used data from the Australian Bureau of Statistics. And to provide non uniform population data, cellular networks were used by Ancel et al., 2017 to create a near-real-time model of the population distribution.

2.2.6 Incident stress model

The incident stress model describes how much harm an individual is exposed to. Where in other term, how protected an individual from getting hit by the debris. Some papers such as Primatesta, Rizzo, and la Cour-Harbo, 2020, Cour-Harbo, 2020,

and Dalamagkidis et al., 2008 to name a few, take into account an attenuating factor based on the sheltering effect to calculate how much kinetic energy a certain building can hold. In the model described by Ball et al., 2012, the amount of KE that is absorbed by the shelter as a result of its deformation, breaking, and movement is then subtracted from the overall risk posed by the system.

Most of the shelter model in the paper who use it is split into four or five different type of building. From no shelter, sparse tree, small building, medium size building, and large building. Where small building typically describe a residential house or a small shop. Medium sized building where it describe a 2 stories building. And large building is from 3 stories and up. Based on investigations undertaken by the Columbia Accident and Investigation Board and the Department of Defense, Melnyk et al., 2014, where it describe a method for calculating the risk from population distribution, most of the shelter are mostly residential structures and commercial buildings.

As previously mentioned each of these shelter types has its attenuating factor, where it is just a coefficient factors to reduce the ammount of kinetic energy. It can be described as an simple proportional equation 2.9 below

$$S \cdot E_k \tag{2.9}$$

Where S is a shelter factor ranging from 0 to 1.0, and E_k is kinetic energy in joule

2.2.7 Harm model

The harm model describe the severity of UAV impact towards the population. The harm model usually measured in probability of fatality given UAV impact to a person. The impact harm model can be categorized based on its severity from minor, major, and fatal injury.As previously mentioned many literature paper use kinetic impact energy as a variable when it comes to determining the fatality imposed by the UAV. Of every injury mechanism, the most common studied harm mechanism is a blunt force trauma. According to Shelley, 2016, the most likely impact injuries are injuries to the head, particularly skull fracture.

For various impact locations, the a charateristic of fatality probability model have been developed as as a function of kinetic energy by Harwick et al., 2007. The model describes the harm response of an average male, averaged over varying impact orientations, as a binomial regression function. Binomial distribution is a function that map a given input in this case kinentic energy to a false or true statement in this case a resulted fatality. Therefore, given a kinetic energy, does that energy can cause fatality to a person.

The binomial distribution is calculated using a sigmoid function, more commonly known as logistic curve, shown in figure **??**. logistic curve probability of fatality

described in Shelley, 2016 is presented in equation **??**. Where threshold is the impact energy associated with a 50% probability of a fatality (measured in Joules), Energy imapct is the impact energy. A number of advancements over this standard equation have been presented by Dalamagkidis et al., 2008 and Shelley, 2016 to name a few.

$$??f(x) = \frac{1}{1 + e^{-\frac{(x - threshold)}{energy\ \ impact}}} \tag{2.10}$$

Figure 2.4: Generelized logistic fatality function given energy impact

Magister, 2010 makes use of a Blunt Criteria (BC) that relates the kinetic energy on impact with the body's ability to tolerate the energy on impact, which is expressed using the Abbreviated Injury Scale (AIS). The Blunt Criterion is a parameter used to assess the severity of blunt force trauma, taking into account the energy transferred during the impact and the body's tolerance to that energy. The Abbreviated Injury Scale is a globally accepted scoring system used to classify and describe the severity of injuries. The AIS assigns a numerical value to injuries ranging from 1 (minor) to 6 (fatal), providing a standardized method to quantify the impact of injuries.

Overall, the use of ground risk models has the potential to significantly improve the safety and effectiveness of UAV operations. By identifying and prioritizing potential risks, ground risk models can help operators make informed decisions about how to mitigate or manage those risks. As the use of UAV continues to grow and

expand into new applications, the importance of ground risk modeling is likely to increase as well.

As an summary, think the ground risk model as a series of queestion asked in order. Where the questions are, how likely the will UAV fail, if the UAV fail where will it land, what does the descent landing looks like in ballistic, fly away, glide, or in parachute mode. If it land how likely is it to hit people in certain area. Are people in the somekind of shelter? if it does how safe is it if the drone hit them, does that hit cauase fatality?

2.3 Pathfinding Algorithms

This study not only for generating the ground risk map for the risk assesment but also measures and finds the path taken by the aircraft to find the least amount of risk and distance traveled required. That is why pathfinding algorithms are essential for UAV navigation, particularly for determining safe and efficient routes. There are multiple pathfinders, or in general, a method that is being used in the UAV industry as a navigation algorithm. All of these algorithms have different criteria, operating principles, and goals to complete. Therefore we are trying to reviews several common pathfinding algorithms, highlighting their advantages and limitations, and explains why the A* algorithm is selected for our study.

2.3.1 Bread-First Search

Bread-First Search (BFS) is a fundamental searching algorithm where it works by starting at the root node and exploring all neighboring nodes. Then, it moves to the nearest nodes and explores their unexplored neighbors, continuing this process until the goal is reached. For a more analogous example, you are at a crowded networking event, beginning at the entrance, and you need to locate a colleague. You initially survey all individuals in the immediate vicinity, then progress incrementally, systematically examining each subsequent group. This methodical approach ensures you comprehensively search each "level" before advancing further.

We can see how the algorithm supposed to "walk" on the simple graph tree on the figure 2.5. Where we start at the left-most node, and it spread between its current neighbor first before increment to the right.

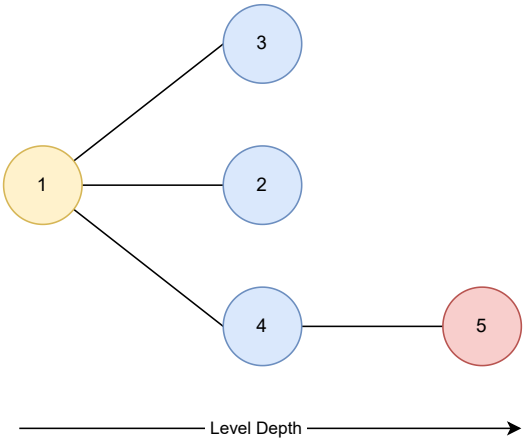


Figure 2.5: BFS graph tree

One of the main advantages of BFS is its guarantee to find the shortest path in terms of the number of edges in unweighted graphs. This algorithm is also straightforward and easy to implement. However, BFS can be very memory-intensive as it needs to store all nodes at the current depth level before proceeding to the next, which can be inefficient for large graphs or when memory resources are limited. Furthermore, BFS does not perform well on weighted graphs where the cost of traversing edges varies significantly Cormen et al., 2022.

2.3.2 Depth-first search

Depth-First Search (DFS) explores as far down a branch as possible before backtracking, making it useful for exhaustive searches of all possible paths. DFS starts at the root node and explores each branch to its fullest before moving to the next branch, we can see the algorithm works on the figure 2.6. This approach can be beneficial in situations where the complete path needs to be explored or specific deep paths are of interest.

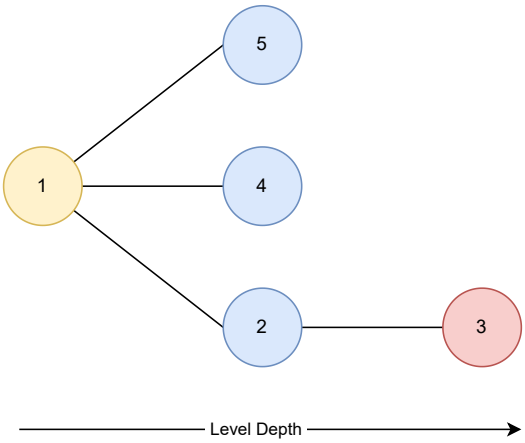


Figure 2.6: DFS graph tree

DFS is less memory-intensive than BFS since it only needs to store the nodes along a single path from the root. It is also simple to implement. However, DFS does not guarantee finding the shortest path. In cases of deep or potentially infinite graphs, DFS can get stuck in long paths, leading to inefficiency. This characteristic makes DFS less suitable for pathfinding where the shortest or optimal path is required Knuth, 1997.

2.3.3 Djikstra's algorithm

Dijkstra's Algorithm is a widely used method for finding the shortest path in graphs with non-negative edge weights. It works by maintaining a set of nodes whose shortest distance from the source is known and repeatedly expanding the shortest known path by one edge. The algorithm uses a priority queue to efficiently select the node with the smallest tentative distance. This algorithm can be represented by equation 2.12 and can be expanded into equation 2.11

$$f(n) = \min\{g(n)\} \tag{2.11}$$

Equation 2.11 is a generalized form of the algorithm, where to find the shortest distance, we must minimized the cost function of $g(n)$. The cost function can be a weight between node connection or the distance between the node. Since we mostly interested in calculating optimum distance, we then expand the cost function to calculate euclidean distance $w(u, v)$ as is cost function. Which resulted in equation 2.12 below

$$d(v) = \min_{u \in V}\{d(u) + w(u, v)\} \tag{2.12}$$

Where $d(v)$ is the shortest known distance from the start node to node v. u is a node in the graph that is adjacent to v. V is the set of all nodes in the graph. And $w(u, v)$ is the weight, or cost, of the edge connecting nodes u and v.

Dijkstra's Algorithm guarantees finding the shortest path in weighted graphs without negative weights, making it highly reliable for a variety of applications. It is efficient when implemented with a priority queue, especially for sparse graphs. However, the algorithm can be computationally intensive for very large graphs, which limits its scalability. For graphs with uniform weights, simpler algorithms like BFS may be more efficient Dijkstra, 2022.

2.3.4 Greedy best-first search

Greedy Best-First Search (GBFS) is an algorithm that uses a heuristic to prioritize nodes that appear to be closest to the goal. Unlike Dijkstra's Algorithm, which considers the cost to reach the current node and the estimated cost to reach the

goal, Greedy Best-First Search only uses the heuristic estimate, as is in equation 2.13.

$$f(n) = \min\{h(n)\} \tag{2.13}$$

Where instead of cost function $g(n)$ we use only heuristic function of $h_(n)$. Heuristic function can be any function that suited the use cases of the algorithm. But when we are discussing about spatial based path finder, we commonly use distances as a cost function.

Even though in a glance that it might seems that this algorithm heuristic function use the same function cost as in the Dijkstra's algorithm. Be aware that in Dijkstra that it use sum of all is traversed node. While in greedy algorithm it only measure the distance between the current node to its final node. Take a look at figure 2.7

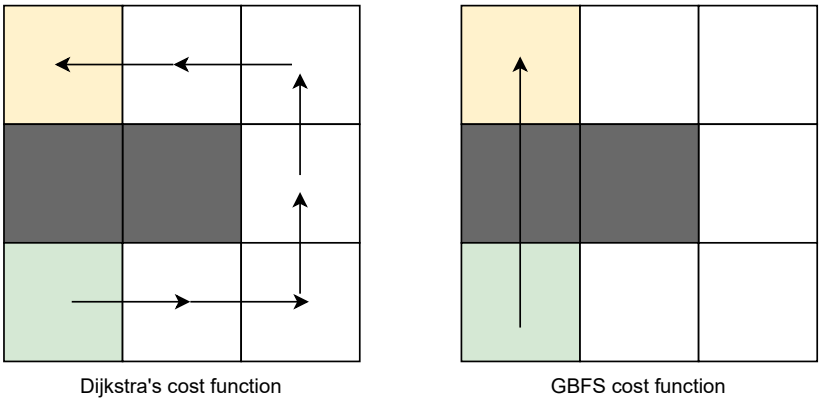


Figure 2.7: Differences of cost function between GBFS and Dijkstra's

As you can see, when there is blocked path between start node colored in green and the end node colored in orange. The cost function of the Dijkstra's algorithm compound its true per step distance, while the GBFS point straight from the start to the end.

GBFS required much lower computation resource and relatively easy to implement. However, this speed comes at the cost of not guaranteeing the shortest path, especially in graphs where the heuristic is not perfectly accurate. This algorithm can be useful when a quick, approximate solution is needed rather than an exact shortest path Russell and Norvig, 2016.

2.3.5 A*

The A* algorithm essentially combines the distance calculation of Dijkstra's algorithm with an additional heuristic function that estimates the remaining distance to the goal, shown in 2.14. This heuristic guides the search process more efficiently

toward the goal, often reducing the number of nodes that need to be explored. This become apparent when exploring very large ammount of grid cells where dijkstra need to explore every cell to determine least ammount of distance, where A* just need to move into the direction of the heuristic vector.

$$f(n) = \min\{g(n) + h(n)\} \tag{2.14}$$

The A* algorithm is a widely used pathfinding and graph traversal algorithm that is known for its efficiency and accuracy. By using a heuristic function, A* often explores fewer nodes than Dijkstra's algorithm, especially in large graphs. A* is guaranteed to find the shortest path if the heuristic function is admissible. An admissible heuristic is one that never overestimates the cost to reach the goal; in other words, it is always optimistic. This ensures that A* will always find the optimal path. However, the performance of A* heavily depends on the quality of the heuristic function. An inappropriate heuristic can degrade the efficiency Knuth, 1997.

An admissible heuristic is one that never overestimates the cost to reach the goal. The Euclidean distance is a common example of an admissible heuristic for pathfinding in a 2D plane because it represents the shortest possible straight-line distance between two points.

For two points A and B with coordinates (x_1, y_1) and (x_2, y_2) respectively, the Euclidean distance $h(A, B)$ is given by:

$$h(A, B) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \tag{2.15}$$

This heuristic is admissible because it never overestimates the true shortest path distance, which would be along the actual path in the grid or graph.

In contrast, a non-admissible heuristic overestimates the cost to reach the goal. For example, let's define a heuristic $h'(A, B)$ as:

$$h'(A, B) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} + 5 \tag{2.16}$$

In this case, $h'(A, B)$ is not admissible because it always overestimates the actual shortest path by an additional 5 cells. This overestimation can lead the A* algorithm to make incorrect decisions about which nodes to explore, potentially causing it to miss the optimal path.

Risk-based A*

Instead of using none weighted graph in a normal A*, we can insert weighted factor in each node. This weight factor is the risk itself, where now not only it tries to minimized the path length but it also tries to find the path with the least ammount of risk. This new function is The Risk-Based A* algorithm which is an extension of the standard A* algorithm. The cost function for the Risk-Based A* algorithm is modified to include a risk component. The cost function $f_{\text{risk}}(x)$ is given by on equation 2.17:

$$f_{\text{risk}}(x) = g(x) + h(x) + R(x) \tag{2.17}$$

where:

- $g(x)$ is the actual distance from the start node to the current node.
- $h(x)$ is the heuristic estimate of the distance from the current node to the goal.
- $R(x)$ is the risk associated with the path from the start node to the current node.

In this formula, $f_{\text{risk}}(x)$ represents the total cost of a path, considering both distance and risk. By minimizing this cost function, the algorithm finds the path that balances the shortest distance with the least risk. Primatesta, Rizzo, and la Cour-Harbo, 2020 and la Cour-Harbo, 2019 use this method to find the overall least risk path with the least possible distance to take to.

2.3.6 Rapidly-exploring random tree

The Rapidly-exploring Random Tree (RRT) algorithm is a widely used method for solving motion planning problems, particularly in high-dimensional spaces and environments with obstacles. It is extensively utilized in robotics for pathfinding and motion planning. The algorithm works by incrementally building a space-filling tree that explores the feasible regions of the state space. RRT is especially advantageous in scenarios where the environment is dynamic and complex, making it challenging to find a path using traditional methods.

RRT operates in a state space, where each state represents a configuration of the system, such as the position and orientation of a robot. The algorithm constructs a tree T that starts at the initial state x_{init} and incrementally expands towards unexplored regions of the state space. Each node in the tree represents a state, and each edge represents a feasible transition between states. The tree is expanded by randomly sampling a state x_{rand} in the state space, finding the nearest node x_{nearest} in the tree to this random state, and then creating a new node x_{new} in the direction of x_{rand} from x_{nearest}.

The RRT algorithm's main steps can be summarized as follows: 1. Initialize the tree T with the initial state x_{init}. 2. Repeat until the goal state is reached or a maximum number of iterations is exceeded:

- Randomly sample a state x_{rand} from the state space.
- Find the nearest node x_{nearest} in the tree T to x_{rand}.
- Create a new node x_{new} by moving from x_{nearest} towards x_{rand}.
- Add the new node x_{new} to the tree T.

Mathematically, the nearest node x_{nearest} is found using the Euclidean distance, show in equation 2.18:

$$x_{\text{nearest}} = \arg\min_{x \in T} \|x - x_{\text{rand}}\| \tag{2.18}$$

where $\|\cdot\|$ denotes the Euclidean norm. The new node x_{new} is then generated by moving a fixed step size δ from x_{nearest} towards x_{rand}, shown in equation 2.19:

$$x_{\text{new}} = x_{\text{nearest}} + \delta \cdot \frac{x_{\text{rand}} - x_{\text{nearest}}}{\|x_{\text{rand}} - x_{\text{nearest}}\|} \tag{2.19}$$

Here, δ is the step size, and the direction vector $\frac{x_{\text{rand}} - x_{\text{nearest}}}{\|x_{\text{rand}} - x_{\text{nearest}}\|}$ ensures the new node is on the line connecting x_{nearest} and x_{rand}. To give more clarity, take a look at figure 2.8 below, where the root node 1 try to reach the final node 4 by spawning node x_{rand} and extends its connection to the nearest node $X_{nearest}$

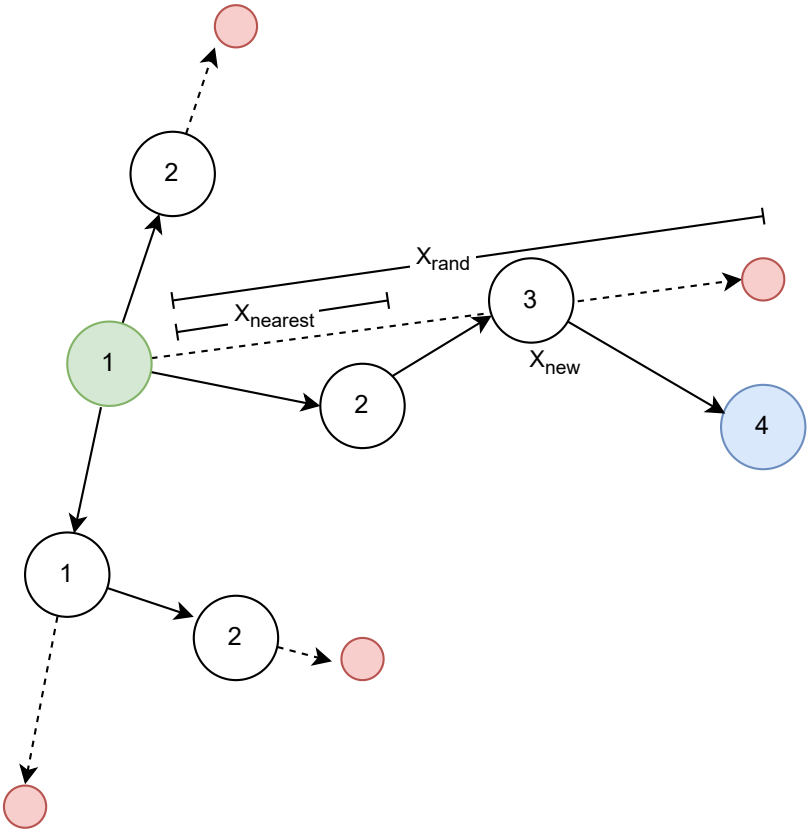


Figure 2.8: Simplified RRT Graph

In layman's terms, the RRT algorithm can be likened to a person exploring a forest. The person starts at a known location (initial state) and randomly picks a direction to move towards (randomly sampled state). They then find the closest point they have already explored (nearest node) and take a step in the direction of the new point (new node). This process is repeated, expanding the area explored until the destination is reached or the exploration is halted.

The RRT algorithm is advantageous because it efficiently explores large state spaces by focusing on unvisited regions. This exploration property makes it suitable for high-dimensional and complex environments. However, RRT may not always find the optimal path, and its performance can be sensitive to the choice of step size δ and the distribution of random samples x_{rand}. Extensions and variations of RRT, such as RRT* and Informed RRT*, have been developed to address some of these limitations by improving path optimality and convergence rates. RRT* works by pruning a longer path that take to the same node, take a look at figure 2.9 where 2 lower node 3 is not pruned in RRT but pruned in RRT*

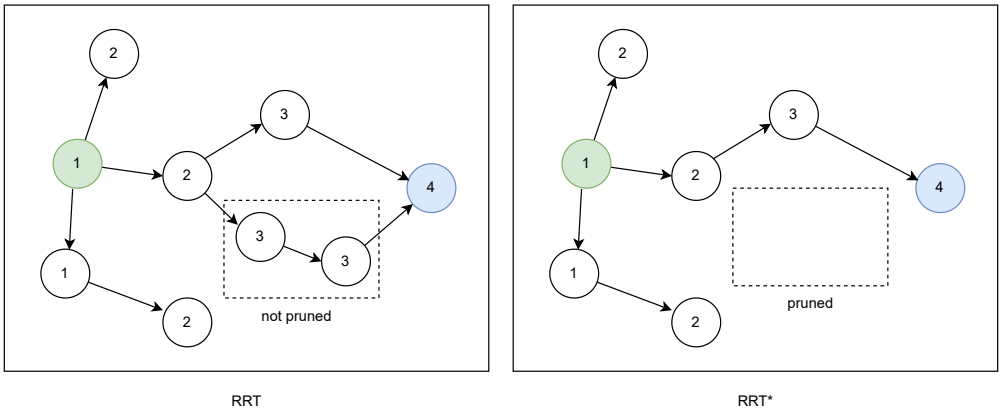


Figure 2.9: RRT vs RRT*

In conclusion, the RRT algorithm is a powerful tool for motion planning in robotics, capable of handling complex and dynamic environments. Its ability to rapidly explore the state space makes it a popular choice for real-time pathfinding applications LaValle, 2006, however because since it excel to find solution fast, it is not guarantee to be optimal. But we are not going to dive deeper into these extension since RRT itself is for real time on the fly path finder, where our paper is for calculating pre-flight ground risk model.

2.3.7 Ant colony optimization

Ant Colony Optimization (ACO) algorithm is a probabilistic technique inspired by the foraging behavior of ants. It is used for finding optimal paths in graphs, and it is particularly effective for solving combinatorial optimization problems such as the travelling salesman problem Dorigo, 2007. ACO simulates the behavior of ants searching for food, where ants deposit pheromones on paths they take, and these pheromones guide subsequent ants towards promising paths.

The algorithm operates in the following steps: 1. Initialize pheromone levels on all edges of the graph. 2. For each iteration, simulate the movement of multiple ants:

- Each ant constructs a solution by moving from one node to another based on the probability influenced by pheromone levels and heuristic information.
- Update pheromone levels based on the quality of the solutions constructed by the ants.

3. Repeat the iterations until a termination condition is met (e.g., a maximum number of iterations or convergence).

The probability $p_{ij}(t)$ that an ant moves from node i to node j at time t is given by equation 2.20:

$$p_{ij}(t) = \frac{[\tau_{ij}(t)]^{\alpha}[\eta_{ij}]^{\beta}}{\sum_{k \in \mathcal{N}_i}[\tau_{ik}(t)]^{\alpha}[\eta_{ik}]^{\beta}} \tag{2.20}$$

where:

- $\tau_{ij}(t)$ is the pheromone level on edge (i, j) at time t.
- η_{ij} is the heuristic information (e.g., the inverse of the distance between nodes i and j).
- α and β are parameters that control the influence of pheromone and heuristic information, respectively.
- $\mathcal{N}_i$ is the set of neighboring nodes of node i.

The pheromone update rule is given by:

$$\tau_{ij}(t+1) = (1-\rho)\tau_{ij}(t) + \Delta\tau_{ij}(t) \tag{2.21}$$

where:

- ρ is the evaporation rate, representing the pheromone evaporation over time.
- $\Delta\tau_{ij}(t)$ is the amount of pheromone deposited by the ants, typically proportional to the quality of the solution.

The ACO algorithm can be likened to a group of ants exploring different paths to find the shortest route to food. As ants travel, they leave a trail (pheromone) that other ants can follow. Over time, the paths with stronger pheromone trails attract more ants, reinforcing good paths and eventually leading to the discovery of the optimal route. Take a look at figure 2.10 below where the first iteration all path has the same ammount of pheromone, but after couple of iterations the shorter path has a much more pheromone since ant agent traverse it more often, where the least likely route has its pheromone evaporated.

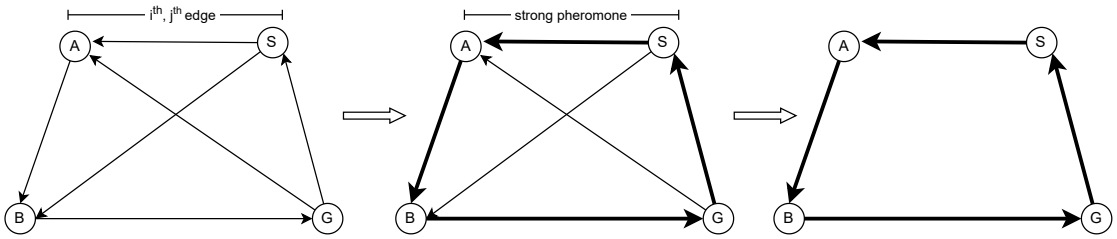


Figure 2.10: Graph with ACO to optimize shortest total route

The Ant Colony Optimization algorithm offers several advantages. It is highly effective for tackling complex optimization problems, making it a versatile tool for

a wide range of applications. The algorithm's ability to find near-optimal solutions within a reasonable timeframe adds to its appeal, especially for time-sensitive tasks. Furthermore, ACO is adaptable to dynamic changes in the environment, allowing it to maintain performance even when conditions shift. However, the algorithm is not without its drawbacks. It can require significant computational resources, particularly for large-scale problems, which may limit its practicality in resource-constrained settings. Additionally like any other complex algorithm with hyperparameters, the performance of ACO is sensitive to parameter settings, necessitating careful tuning to achieve optimal results. There is also a risk of converging to a suboptimal solution if the algorithm is not properly managed, highlighting the importance of adequate oversight and control Dorigo, 2007.

Although the ACO algorithm is a robust and flexible optimization technique that mimics the natural behavior of ants to solve complex problems. The problem it tries to solve is not suitable for finding ground risk model. Remember that ACO is try to spread the ant to explore every possible cells to determine the shortest distance to cover every cell.

2.3.8 Generative pathfinding AI

Generative Pathfinding AI algorithms leverage advanced machine learning techniques, particularly generative models, to discover optimal paths in complex environments. These algorithms utilize neural networks, such as Generative adversarial networks (GANs), to generate feasible and efficient paths. By learning from data, these models can generalize to new, unseen environments and provide robust pathfinding solutions Silver et al., 2016.

The algorithm operates in the following steps: 1. Train a generative model on a dataset of known paths. 2. Use the trained model to generate new paths by sampling from the learned distribution. 3. Evaluate the generated paths based on specific criteria (e.g., shortest distance, minimal risk).

Generative Pathfinding AI offers several significant advantages. The most notable is its capability to learn and generalize from vast amounts of data Russell et al., 2010, enabling it to adapt to complex and dynamic environments efficiently. This adaptability makes it suitable for scenarios where the environment is continually changing or unpredictable. Additionally, it can efficiently generate multiple path solutions, providing a variety of options to choose from, which is particularly useful in decision-making processes Russell et al., 2010. However, these advantages come with some challenges. Generative Pathfinding AI requires large amounts of training data to achieve high performance, which may not always be readily available. The training process itself is computationally intensive, demanding substantial computational resources and time. Moreover, the paths generated by the model need to be validated for feasibility, as not all generated paths will be practical or optimal. This validation step is crucial to ensure the reliability of the solutions provided by the AI.

In summary, Generative Pathfinding AI represents a cutting-edge approach to pathfinding, leveraging the power of machine learning to discover efficient paths in complex environments. Its ability to learn and adapt makes it a promising tool for a wide range of applications.

2.3.9 Justification for choosing risk-based A*

Among the algorithms reviewed, A* stands out due to its balanced approach combining the benefits of Dijkstra's Algorithm and Greedy Best-First Search. A* is particularly advantageous for pre-operational UAV path planning as it ensures the optimal path in terms of both distance and risk. By incorporating a risk coefficient into the cost function, the Risk A* variant can balance the need for short paths with the requirement to minimize exposure to hazardous areas. Studies by Primatesta, Rizzo, and la Cour-Harbo, 2020 and Clothier et al., 2007 have demonstrated the effectiveness of the Risk A* algorithm in developing ground risk management models, further supporting its suitability for this application.

While generative pathfinding AI algorithms offer advanced capabilities by learning from vast amounts of data and adapting to complex environments, they come with significant drawbacks. The training process for these models is prohibitively expensive, requiring substantial computational resources and time. This makes them less practical for applications where resources are limited or quick deployment is necessary.

In conclusion, while various pathfinding algorithms offer different benefits, A* provides the most balanced and optimal solution for UAV navigation in terms of minimizing both distance and risk, making it the preferred choice for our study.

CHAPTER 3
BACKGROUND THEORY AND METHODOLOGY

3.1 General Overview

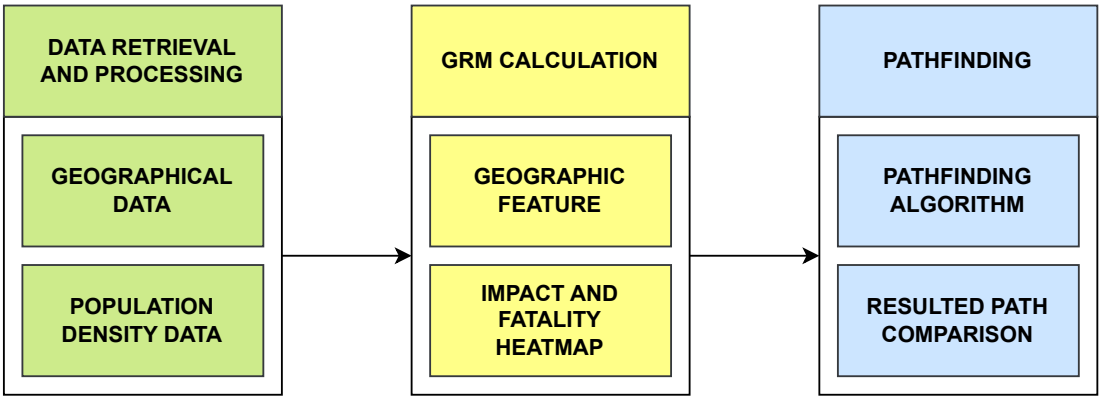


Figure 3.1: General overview

In general, there are three main objectives to create an optimal risk-aware flight path. These are data retrieval and processing, ground risk map calculation, and generating the most optimal path using the pathfinder algorithm.

To collect necessary data, we are using free services provided by OpenStreetMap (OSM) "OpenStreetMap Wiki", 2022 to retrieve geographic data, and Global Human Settlement Layer (GHSL) "Global Human Settlement - Mission - European Commission", 2023 to retrieve population density throughout the earth.

The Ground Risk Model (GRM) is calculated using geographic and population data, and also the type of aircraft that are being used in the calculation. The result would be descent location, impact energy, and fatality probability heat map.

After we compute and discretized the GRM. We find the optimal path with the least amount of risk using pathfinding algorithms, where the result would be compared.

3.2 Prerequisite data

3.2.1 Online geographic data

OSM "OpenStreetMap Wiki", 2022 is a collaborative project that aims to create a free, editable map of the world. It was founded in 2004 and has since grown to become one of the largest and most popular sources of geospatial data.

OSM is built on the idea that anyone can contribute to the map by adding and

updating data. Users can add data using Global Positioning System (GPS) devices, aerial imagery, or by manually tracing over satellite imagery. The data is then stored in a database and made available under an open license, allowing anyone to use and share the data.

There are multiple ways to use the OSM service. To retrieve the geographic data we can send the querry to Overpass Application Programming Interface (API) and recieve the defaulted Geographic JavaScript Object Notation (geoJSON) data. OSM also provide geocoding and reverse geocoding using Nominatim API.

3.2.2 Population data

GHSL "Global Human Settlement - Mission - European Commission", 2023. It is a dataset produced by the JRC (Joint Research Centre) of the European Commission that offers details on the distribution and location of human settlements worldwide.

The GHSL classifies and examines patterns of urbanization and human settlement using new spatial data mining technologies and remote sensing data, such as satellite images. The dataset is periodically updated to reflect changes in urbanization and population growth and contains data on built-up areas, population density, and other indices of human presence.

Overall, the GHSL is a valuable tool for comprehending the spatial dynamics and patterns of human settlement and can assist planners in making better-educated choices concerning urbanization and sustainable development.

3.2.3 Geographic Information System

Geographic and population data then stored and processed by Geographic Information System (GIS), tools used for storing, manipulating, and visualizing geographic data. They allow users to analyze and understand spatial relationships, patterns, and trends in geographic data.

GIS consists of several key components:

- Data: GIS uses data that is linked to a geographic location. This data can be in the form of points, lines, or polygons (areas) and can include attributes such as population, land use, or elevation.
- Hardware and software: GIS requires specialized hardware and software to store and analyze the data. This includes computers, servers, and specialized software such as ArcGIS or QGIS.
- Maps: GIS allows users to create interactive maps that can be used to visualize and analyze the data. Maps can be customized with different layers of information, such as roads, land use, and population density.
- Analysis: GIS allows users to perform spatial analysis on the data, such as

calculating the distance between two locations or identifying patterns in the data.

There are multiple GIS software suites, from standalone programs such as ArcGIS, and QGIS, to package base library for python, such as GeoPandas, Rasterio. In this paper we are going to use GeoPandas and Rasterio and its supporting libaries.

3.3 Risk map

The Risk map is a graphic representation depicting a potential risk associated with the UAV operation. Risk maps are divided equally in distance into cells. Each cell represented a bounded geographical location. The map can be represented by a $M \times N$ size matrix, which each $M \times N$ is the number of cells in each row and column respectively. Each $\mathbf{R}(i, j)$ cell has centroid coordinates which represented the geographic location of (x, y) in any coordinate reference system Coordinate Reference System (CRS).

If we use a slight notation abuse, which is $\mathbf{R}(i, j)$, each cell coordinate (i, j) is represented as a geographical location in which case shifted by x and y amount. Figure [Dex] illustrates the Risk map matrix.

3.3.1 Georeference layers

In a sense risk map is constructed by stacking layer of features, each layer itself is obtain by analyzing each geographical feature and population density to calculate the ground risk map. In this paper, multilayer frameworks are used based on the work of Primatesta, Rizzo, and la Cour-Harbo, 2020. As the name implied, this framework contains multiple georeference layers. These layers are :

- **Population density layer**: Describes population density in the area.
- **Obstacle layer**: Describes geographic features that can block aircraft flight paths such as building height, or any other obstacle.
- **Sheltering factor layer**: Describes geographic features that provide shelter for humans in the area.
- **No-Fly zone layer**: Describes geographic features in which aircraft cannot enter, such as military base, or airport

3.3.2 Population density layer

The population density layer describes the population density and distribution in the area. It is one of the most crucial factors in the risk assessment is population density since it indicates how many individuals can be affected by a vehicle crash and how they are dispersed over the map.

To conduct an accurate risk assessment it is necessary to provide population density data with fine resolution. since the likelihood of colliding with a person on the ground is particularly impacted by population density.

The population data are frequently gathered by the government human settlement institute. However, this approach is not easily ported from country to country since different institutes produce population data differently. As previously mentioned above GHSL would be used in this paper, since it already has the global database.

In this study, the population density layer is represented by a 2D georeference map with each cell corresponding to the value of population per cell area size with cell area in unit m^2, therefore $people/m^2$. Logically, the map would be in matrix form, $\mathbf{D}$, with each cell element of $\mathbf{D}(x, y)$.

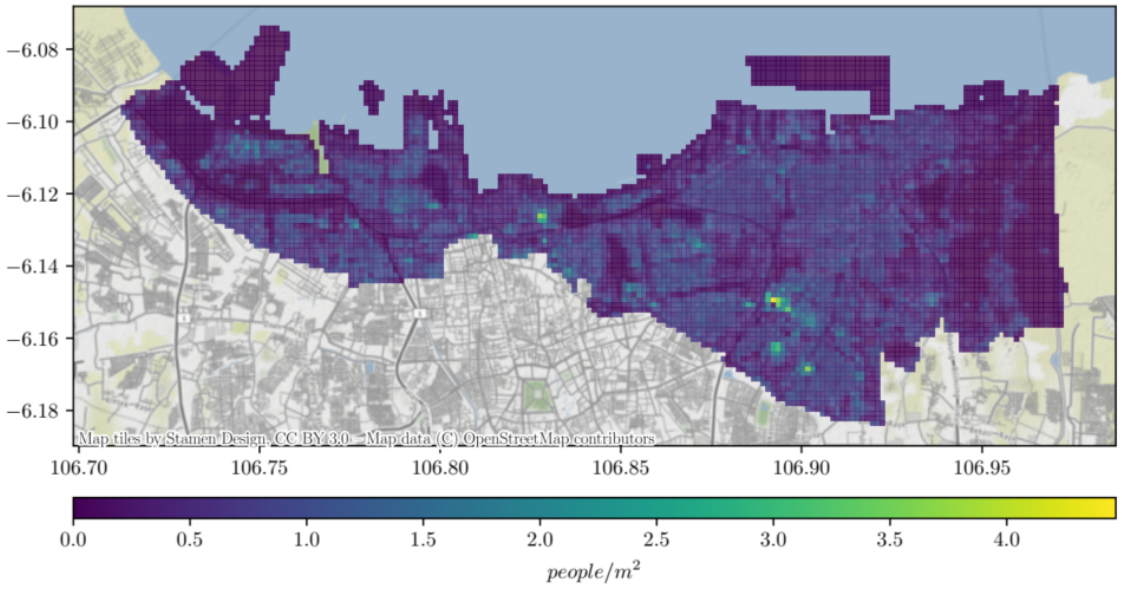


Figure 3.2: Population density map

3.3.3 Obstacle layer

Obstacle layer describe which cell has contains a building or geographic feature which higher than flight altitude of UAV. the data can easily be gathered by filtering, and mark it as impasable terrain. since it is just another georeference layer, we can represent the as a matrix $\mathbf{O}(x, y)$ which describe maximum height of building in a given cells.

Noted that this layer does not explicitly accounted into risk factor analysis, but instead only for determining each cell is impasable for path finder.

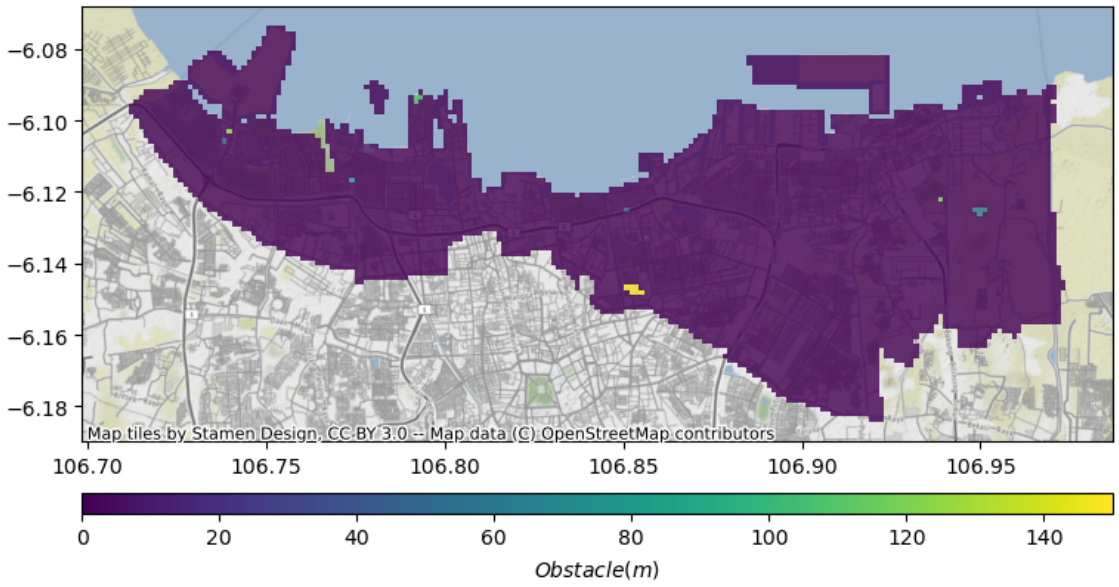


Figure 3.3: Height obstacle map

3.3.4 Sheltering factor layer

Sheltering Factor layer is a term for describing, the layer that focuses on assessing the presence and characteristics of physical structures or objects that can provide shelter. It is represented with a matrix $\mathbf{S}$, where each element $\mathbf{S}(x, y)$ assumes the sheltering factor value of the corresponding location.

The sheltering factor layer takes into account factors such as the size, shape, material, and location of the structures or objects in the drone's operating environment. It aims to assess how these factors can influence the risk of fatality in case of failure.

Unfortunately, it is impractical to evaluate the sheltering element in full detail in a study of this kind because it is a very complex scenario. The dynamic of drone collision is a complication in itself, particularly since detailed physical building features, like material or wall thickness, are recorded differently around the world.

It is worsened by the fact that there are multiple studies with conflicting results when calculating the absorption of shelter since there is no consensus when defining the sheltering factor. Some studies by Dalamagkidis et al., 2008 and Primatesta, Rizzo, and la Cour-Harbo, 2020 use a scale factor from 0 to 1, or 1 to 10 respectively, representing the energy absorption factor of a given building. While Melnyk et al., 2014 uses a study conducted by the United States Department of Defense (DoD) Harwick et al., 2007 regarding explosive study to measure the energy absorption capability.

Even though it is tempting to use the method given by Melnyk et al., 2014 and DoD since it has the most accurate data. The study conducted by them is more suitable for singular operation-specific evaluation i.e. best case use for generating

event tree, and not suited for generating risk map. There are also multiple problems when using this method, such as the lack of accurate data on building material geographic maps, especially when OSM do not provide a building material data, which would result in inaccurate measurement.

From the previous statement, in this paper, we would use the scale factor proposed by Klepeis et al., 2001 and Primatesta, Rizzo, and la Cour-Harbo, 2020 as a foundation for sheltering factor risk generation. Since it provides a general method to calculate a risk factor. Table 3.1 is the shelter factor for types of features.

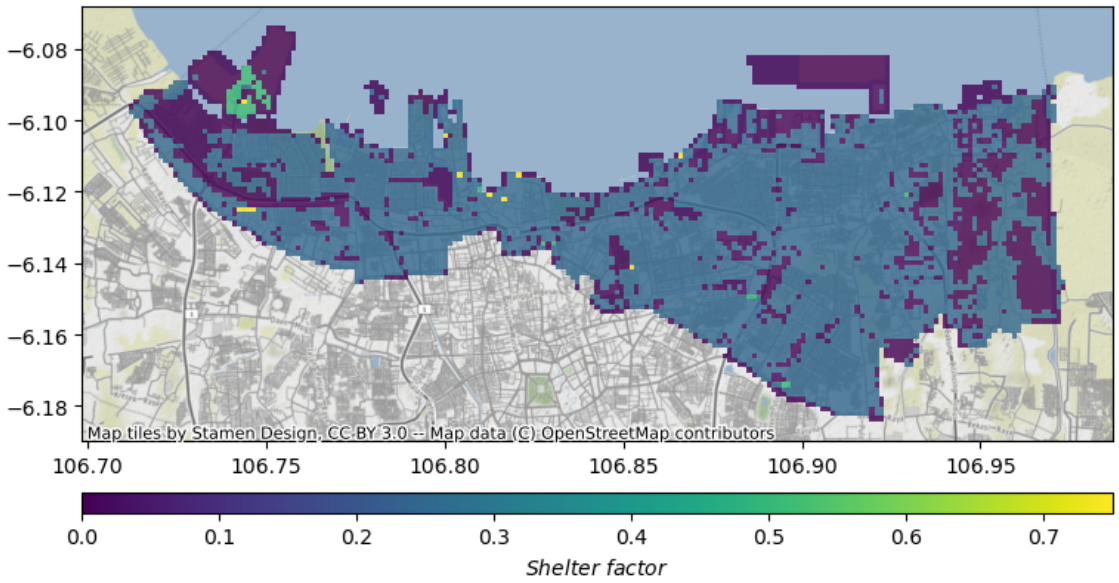


Figure 3.4: Shelter factor map

Sheltering Factor	Type
0	No shelters
0.25	Sparse trees
0.5	Low building
0.75	High building
1.0	Industrial building

Table 3.1: Sheltering factor

3.3.5 No-fly zone layer

The No-fly zone layer refers to a component of a ground risk map or airspace map that delineates areas where drone flights are prohibited. The restrictions in these zones can include areas such as:

- Regulation agency-defined areas, i.e.Federal Aviation Administration (FAA), and International Civil Aviation Organization (ICAO). They restrict airports and military areas.
- Nature-sensitive areas, such as National Park where flying is restricted.
- Safety and security areas, very crowded public areas, such as the city center, or events.
- Operator-defined areas, where areas are restricted by the operator itself for various reason.

In this paper, the no-fly zone layer is a georeferenced map similar to the obstacle layer. And generally work in the same method to obstacle layers. The No-fly zone layers explicitly prohibit the flight of UAV by using two different values, -1 if flying is forbidden, and 0 if flying is allowed. The no-fly zone layer is represented by matrix $\mathbf{F}(x, y)$ defined as:

$$\mathbf{F}(x,y) = \begin{cases} -1 & \text{if flight is prohibited.} \\ 0 & \text{if flight is allowed} \end{cases} \tag{3.1}$$

3.4 Ground Risk Assessment

In this research, the risk is described as the potential to cause harm of UAV to the population on the ground. quantified as the number of fatalities per hour of flight. The risk model is built upon a series of consecutively occurring events, each with its probability. These events are denoted as P_{event}, P_{impact}, and $P_{fatality}$.

$$P_{casualty}(x) = P_{event}(x) \cdot P_{impact}(x) \cdot P_{fatality}(x) \tag{3.2}$$

P_{event} represents the probability of the UAV losing control and descending into the ground. This paper considers four types of events: Ballistic, Unpowered Glide, Flyaway, and Parachute. Mathematical models are utilized to determine the probable impact area for each type of descent. It is important to note that these events are calculated independently of each other. Once the probabilities for each event are computed, they are summed up for each cell, resulting in the probability of the drone impacting that particular cell. This probability is described using a 2D Probability Density Function (PDF). For further information on the descent models, please refer to Sections:3.4.2. The two-dimensional PDF is utilized to calculate the probabilities of P_{impact} and $P_{fatality}$, considering factors such as population density, sheltering factor layer, and impact velocities.

After calculating the UAV crash location following an uncontrolled descent, we need to calculate the Pimpact which indicates the likelihood of the UAV strikes individuals. Where the population density and the impact exposure area can have its influence. And finally $P_{fatality}$, which denotes the probability of fatal injuries

occurring after a person has been impacted. It depends on the kinetic energy at the time of impact and the sheltering factor.

The overall probability of casualty, $P_{casualty}(x)$, is then computed. The $\mathbf{R}$ is a matrix containing calculated of said $P_{casualty}(x)$ in every element of $\mathbf{R}(x, y)$. This approach allows for the assessment of risk associated with UAS flights by considering various factors and probabilistic models for different events during the flight.

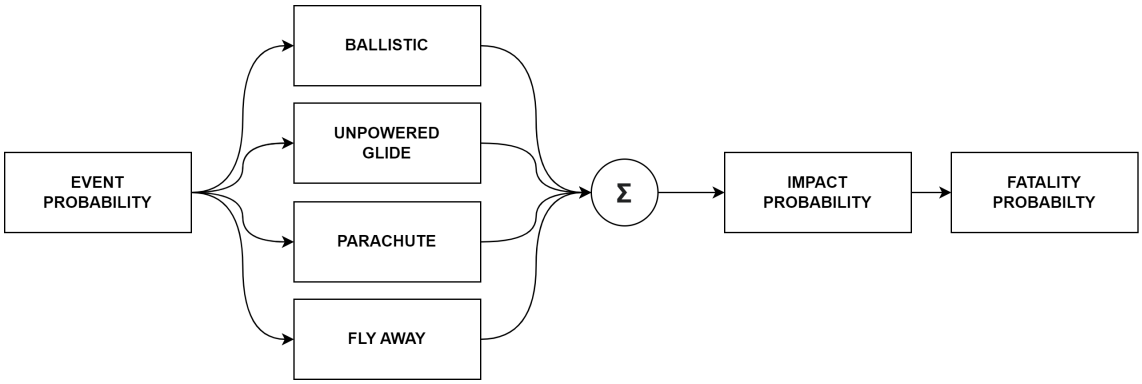


Figure 3.5: General risk assesment flowchart

3.4.1 Failure rate

The failure rate is a metric used to measure the reliability or average failure rate of a system or component. In this study, the MTBF would represent the failure rate. Where MTBF stands for Mean Time Between Failures. MTBF represents the average amount of time that elapses between consecutive failures of a system or component.

MTBF is often expressed in time units, but usually it measures in hours, however, it can also be measured in other such as days or years. A higher MTBF value indicates a longer expected time between failures, suggesting a more reliable system or component.

MTBF is typically calculated using historical failure data. Where this study using the assumption from, Shelley Shelley, 2016 and UAS Task Force **noauthor˙unmanned˙2015** is set into 100 hours.

3.4.2 Descent Event

Information on how the UAV descended into the ground is required for ground risk management. The impact area and the kinetic energy at impact are determined according to the type of descent, which affects how the UAS falls to the ground.

The shape of a UAV has a great influence on how it falls, with fixed wings, generally glides down. While the multirotor fell following the ballistic trajectory. Both of which will influence the falling probability heat map.

These models below are employed to calculate a two-dimensional PDF that quantifies the likelihood of the UAS impacting the ground. The PDF takes into account uncertainties in the initial velocities (horizontal and vertical), drag coefficients, flight altitude, and flight direction.

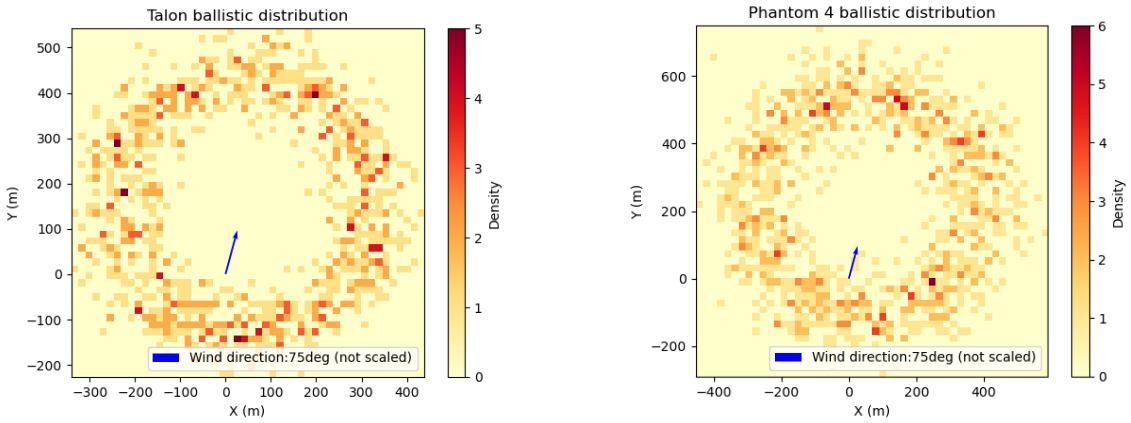
Furthermore, the model generates a two-dimensional matrix containing the estimated horizontal and vertical impact velocities. These velocities are instrumental in computing the probabilities of impact (P_{impact}) and fatality ($P_{fatality}$).

3.4.3 Ballistic Descent

The occurrence of ballistic descent arises when the aircraft experiences a catastrophic failure, such as a structural failure that results in the loss of most of its lift. Consequently, the vehicle undergoes a descent solely driven by the forces of gravity and drag.

$$m\dot{v} = m\mathbf{g} - c \cdot A|\mathbf{v}|\mathbf{v} \tag{3.3}$$

The ballistic descent model used in this work is based on the standard second-order drag model. The model takes into account factors such as the mass of the UAV (m), a drag constant (c), area(A), air density(ρ), gravitational acceleration (g), and the vehicle's velocity vector (v). A study done by Cour-Harbo, 2020 provides greater details of the model that also considers uncertainties and variations in wind conditions.



(a) Fixed wing ballistic descent

(b) Multirotor ballistic descent

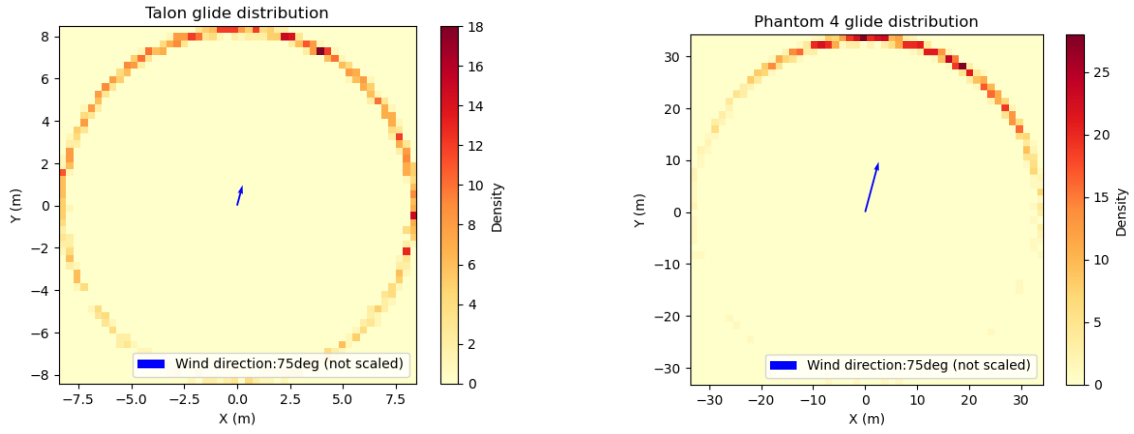
Figure 3.6: Ballistic descent by aircraft type

3.4.4 Glide Descent

Uncontrolled glide may arise depending on the configuration of the aircraft. It is an event when UAV flight profiles are governed solely by the glide ratio or autorotation angle. This event could happen when aircraft loses its power but still retains most of its lift capability.

In the case of a fixed-wing aircraft, this event arises from a loss of thrust or power affecting the flight control surfaces. For a single rotorcraft, it occurs when there is a loss of thrust on the main rotor, causing the aircraft to descend using autopiloted autorotation. However, multi-rotor cannot enter unpowered glide, instead, it enters ballistic descent.

During the uncontrolled glide event, the horizontal distance traveled is computed using the simple formula $dist(h) = \gamma h$, where h represents the flight altitude and γ represents the glide ratio



(a) Fixed wing glide descent

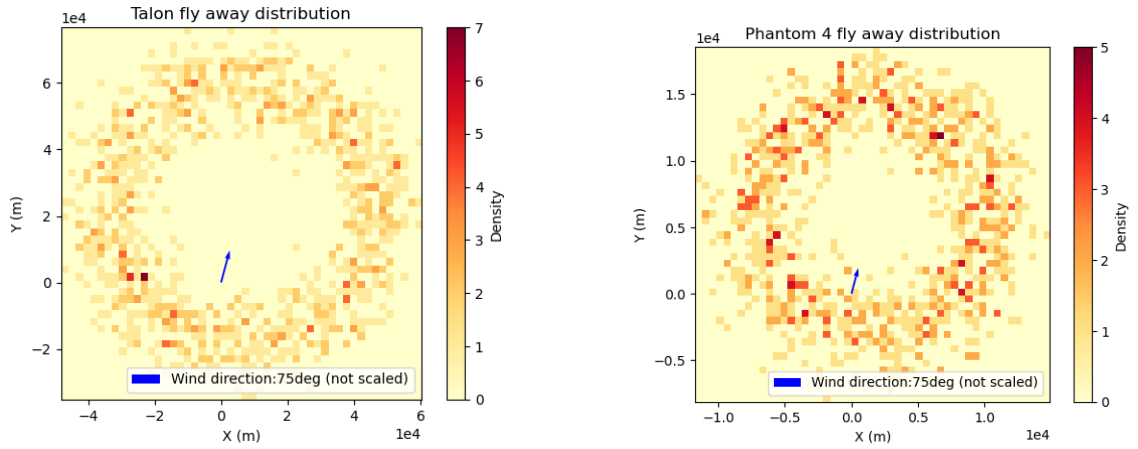
(b) Multirotor glide descent

Figure 3.7: Glide descent by aircraft type

3.4.5 Fly away

The fly-away event occurs when the operator loses complete control over the UAV, while the onboard flight control maintains the stability of the vehicle. As a result, the UAS can fly in any direction until it crashes on the ground. This event is modeled based on a reference paper la Cour-Harbo, 2019 , which proposes a model comprising two components. Firstly, the probability of ground impact decreases linearly with the distance from the UAS position until reaching the maximum distance the vehicle can travel. Secondly, the vertical motion of the vehicle is taken into account, with a higher probability of ground impact near the UAS position, modeled as a normal distribution centered around the UAS position. These two contributions are combined linearly using a scaling factor.

In this study, equal probabilities are assumed for both scenarios to be conservative. It is assumed that the fly-away event terminates with an uncontrolled glide descent. Unlike other descent events, the fly-away event can involve motion in all directions. Figure 16 presents the results of the fly-away event.



(a) Fixed wing fly away descent

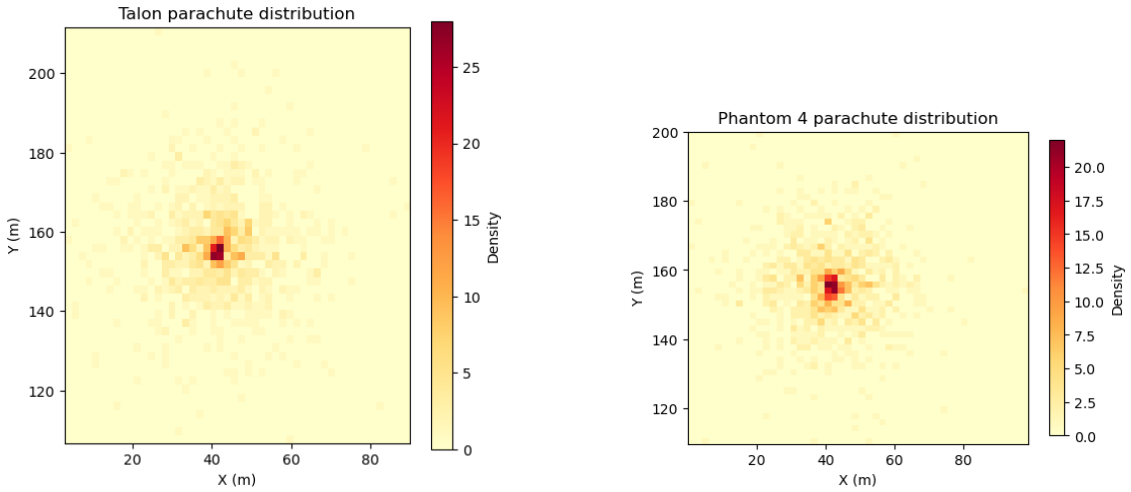
(b) Multirotor fly away descent

Figure 3.8: Fly away descent by aircraft type

3.4.6 Parachute

The parachute descent starts when the UAV flies downward while the parachute is fully extended. The power plants are typically shut off and the parachute is opened when a problem occurs. After the emergency detection and the parachute fully opening, there is, however, a small delay. The only factors affecting the drop at this point are the parachute's aerodynamic features, which are intended to slow the vertical speed.

To model this event, a methodology described in a referenced paper la Cour-Harbo, 2019is employed. The authors of the paper consider factors such as the mass of the vehicle, as well as the physical properties of the parachute, including its area and drag coefficient. Notably, the presence of wind significantly alters the descent dynamics, affecting the direction and velocity of the descent.



(a) Fixed wing parachute descent

(b) Multirotor parachute descent

Figure 3.9: Parachute descent by aircraft type

3.4.7 Wind effect

The wind is one of the important variables in creating a ground risk map since wind speed and direction is determined the impact location of each descent event. Where the wind data can be received from weather forecast information before the operation.

Noted that this study assumes and applies constant wind speed and direction when simulating the descent event. This assumption arose when the wind effect usually encompasses a large local area which is often greater than medium to micro-sized UAV. Added to that, the wind only affected only the descent event and not when the drone is normally operating.

3.4.8 Probability of Impacting an Individual

The P_{impact} is the probability of a UAV striking individual. This parameter is important because, when used in conjunction with the population density parameter previously discussed, it determines the number of people exposed to risk on the ground in the event of an impact.

$$P_{impact}(x, y) = \rho(x, y) \cdot A_{exp} \tag{3.4}$$

The P_{impact} equation describe the population density ρ and exposure area Aexp

Similar to sheltering factor, there is no standard approach when calculating the affected area. However, there are two distinct approaches that most researchers find useful, the empirical method and the geometric method.

Empirical methods are the method that uses regression to calculate impact area

based on the size or the weight of an aircraft. While the geometric method uses the general dimension of the aircraft to calculate the impact area.

Research done by Melnyk et al., 2014 uses the regression method from Ale and Piers, 2000, based on the accident data of large aircraft to calculate the impacted area. However, this regression is created using aircraft weight of more than 1500kg. This can result in underestimating the lethality of common UAV operations which weigh less than 500kg.

A study done by Aalmoes et al., 2015 proposes two types of solutions. empirical regressionand geometric approach. Which stated that, above 500kg the calculation should use the empirical method, or geometric approach if less than 500kg. The empirical data obtained by the Netherland Aerospace Laboratory, when researching the helicopter crashes behavior, resulted in 3.6 relation. While the geometric approach 3.5 is using human impact model.

$$A_{exp}(m) = 230 \ln\left(\frac{m}{1000}\right) + 330 \tag{3.5}$$

$$A_{exp}(\theta) = 2(r_p + r_{uav})\frac{h_p}{\tan(\theta)} + \pi(r + r_p)^2 \tag{3.6}$$

Where A_{exp} is in m^2, with m is kg, θ is impact UAV impact angle. And r_{uav} is UAV radius , h_p is person height, and rp is person radius all in m,

3.4.9 Probability of Fatality

The probability of fatality, denoted as $P_{fatality}$, represents the likelihood that an impact on a person will result in a fatality. Calculating this probability is challenging due to the various complexities involved in an impact event. The human body reacts differently to different types of impacts, and UAV crashes can involve factors such as blunt impact, lacerations, or chemical explosions caused by the battery. As a result, variables like velocity and mass do not have a simple correlation to the severity of injuries, as the human body's response varies depending on the specific body part affected. This lack of consensus on the calculation method for fatality probability arises from the difficulty of accounting for all the intricacies and variations in impact events. However, it is generally accepted to correlate kinetic impact energy to measure the $P_{fatality}$.

$$P_{fatality} = \frac{1}{1 + e^{-k(E_{imp} - E_{thres})}} \tag{3.7}$$

3.8 is a generalized equation based on the logistic function, where the function maps a value of impact energy Eimp to the probability of fatality, with Ethres is an energy value that has a probability to cause fatality above 50%. On average, 58 joules of energy can cause a fatality based on the table below.

Information	Value	Unit	Source
Energy for hazardous debris	45	Joule	Cole et al., 1997
Energy required by fragment for 90% probability of fatality	120	Joule	Cole et al., 1997
Energy required by fragment to be hazardous to human	58	Joule	Gonzales and Murray, 2017

Table 3.2: Energy correlate to fatality

This study would be using more comprehensive $P_{fatality}$ Equation 3.8, with α as threshold energy to cause 50% probability of fatality in a medium level of shelter $S \geq 0.6$, and β is the threshold energy needed to cause fatality when in a non-sheltered area ($S \leq 0.1$).

$$P_{fatality}(x, y) = \frac{1}{1 + \sqrt{\frac{\alpha}{\beta}\left[\frac{\beta}{E_{imp}(x,y)}\right]^{\frac{1}{4S(x,y)}}}} \tag{3.8}$$

The kinetic energy is computed as:

$$E_{imp}(x, y) = \frac{1}{2} m \cdot v_{imp}(x, y)^2 \tag{3.9}$$

Equation 3.8 is based on Equation 3.8 from a study conducted by Dalamagkidis et al., 2008 and Primatesta, Rizzo, and la Cour-Harbo, 2020. Although it is not accounted for chemical and thermal factors as in Melnyk et al., 2014 and Harwick et al., 2007, we argue that since both Melnyk et al., 2014 and Harwick et al., 2007 only suitable for evaluating per single event and not for generating ground risk map.

3.5 Finding optimal path

After creating a risk map, the next objective is to find the optimal flight path with the least amount of risk. From a naive method point of view, computing flight path is quite easy since 99% of the operation is only a straight path from point A to point B since there is little to no obstacle. Instead, the risk factor has a bigger impact on the Pathfinder compared to real physical obstacles.

There are many types of pathfinder algorithms with each of its advantage, disadvantage, and its most effective usage of operation. In this paper, we would be using A* and Dijkstra's algorithm to find the optimal path and compare each algorithm. There is an explanation in section [A] for reasons why we are not going to use other algorithms.

3.5.1 Problem statement

Given:

- A grid with dimensions $M \times N$, where each cell represents a location in the grid.
- A start cell S and a goal cell G in the grid.
- Each cell in the grid has an associated risk factor (a non-negative real value) that represents the level of risk or danger associated with that cell.

Objective: Find the shortest path from the start cell S to the goal cell G that minimizes the cumulative risk factor along the path.

Formally, let::

- Let $G = (V, E)$ be the graph representation of the grid, where V represents the set of cells and E represents the set of edges connecting adjacent cells in the grid.
- Let $w(u, v)$ be the weight or cost associated with an edge (u, v) in E. In this case, the weight $w(u, v)$ represents the risk factor of moving from cell u to cell v.
- Let $d(u)$ represent the shortest path distance from the start cell S to cell u.

The problem is to find the shortest path $P = (S, v1, v2, ..., G)$ that minimizes the cumulative risk factor along the path, which can be represented by the sum of the weights along the path:

$$min \sum w(v_i, v_{i+1}) \tag{3.10}$$

subject to:

- P is a valid non negative weighthed path from S to G in the graph G.
- $d(G)$ is minimized.

The objective is to find the path P that minimizes the cumulative risk factor, taking into account the risk factors associated with each cell in the grid.

This problem can be solved using various algorithms such as Dijkstra's algorithm, A*, or even extensions like Risk-Aware A* that consider risk factors in the pathfinding process.

Noted that the specific formulation and constraints of the problem may vary depending on additional considerations such as movement constraints, specific risk factor calculations, and other domain-specific requirements. The above problem statement provides a general framework for finding the shortest path based on risk factor in a grid system.

Graph Before using any pathfinder algorithm, the grid map must be defined in the graph. In computer science and mathematics, a graph is a data structure that represents a collection of interconnected nodes, called vertices, (V), and the relationships between them, called edges, (E).

A graph consists of two main components: Vertices (Nodes): These are the fundamental units of a graph and represent entities or objects. Each vertex is typically labeled with a unique identifier or value. Edges: These represent the connections or relationships between vertices. An edge connects two vertices and can be either directed or undirected.

A directed edge has an orientation or a specific direction, while an undirected edge does not. Edges can also have weights or costs associated with them to represent the strength of the relationship or distance between vertices.

A grid typically falls into the category of a regular graph or a lattice graph. A regular graph is a graph in which every vertex has the same number of neighbors. In a grid, each cell is typically connected to its adjacent cells, resulting in a regular graph structure. The number of neighbors depends on the dimensionality of the grid.

The grid system, with its regular arrangement of cells and connectivity between adjacent cells, can also be considered a form of lattice graph. A lattice graph is a specific type of regular graph that forms a lattice-like structure. A lattice graph consists of a regular arrangement of vertices in a grid-like pattern, where each vertex is connected to its adjacent vertices.

3.5.2 Dijkstra's algorithm

Dijkstra's algorithm is a well-known and widely used algorithm for finding the shortest path between a given source node and all other nodes in a graph. It guarantees optimality, meaning it always finds the shortest path, as long as the graph has non-negative edge weights.

Dijkstra's algorithm uses a cost function to calculate the shortest path from a given start node to all other nodes in a graph. The distance in Djikstra's case is represented as a cost function of $g(x)$.

$$g(x) = min(g(x_{goal}), g(x_n)) \tag{3.11}$$

The function $g(xn)$ represents the shortest distance from the start node to a

given node x_n in the graph. It is initialized to a very large value, often infinity, for all nodes except the start node, which is assigned a distance of 0.

$$g(x_n) = \sum_{i=1}^{n-1} g(x_{i-1}) + c(x_{n-1}, x_n) \tag{3.12}$$

where, $\sum_{i=1}^{n-1} g(x_{i-1})$ is all combined cost from a start node to the current node, and $c(x_{n-1}, x_n)$ is a risk-weighted cost function, which describes to

$$c(x_{n-1}, x_n) = \frac{r(x_{n-1}) + r(x_n)}{2} \cdot d(x_{n-1}, x_n) \tag{3.13}$$

where the risk r_x is the average between two nodes, and $d(x_{n-1}, x_n)$ is an euclidean distance between two nodes.

The algorithm iteratively selects the node with the minimum distance among the unvisited nodes and updates the distances of its neighboring nodes if a shorter path is found. This process continues until all nodes have been visited or until the goal node is reached, depending on the specific problem requirements.

By continuously updating the distances of nodes based on the weights of the edges, Dijkstra's algorithm ensures that it explores paths with lower total costs, gradually converging towards the shortest path from the start node to each node in the graph.

3.5.3 A* algorithm

A* algorithm is an extension of the Dijkstra algorithm and uses a heuristic function to guide the search. From 3.14 we can see that the addition of heuristic function $h(x_n)$. Noted that the implementation of $g(x)$ is similar to previously mentioned in 3.12.

The cost function can be represented as:

$$f(x) = g(x_n) + h(x_n) \tag{3.14}$$

where:

- $f(n)$ is the total estimated cost of node n,
- $g(n)$ is the actual cost from the start node to node n,
- $h(n)$ is the heuristic estimate of the remaining cost from node n to the goal node.

Algorithm 1 Dijkstra's Algorithm

```
1: function Dijkstra(start, goal)
2:     distances ← {}
3:     previous ← {}
4:     for all node in graph do
5:         distances[node] ← ∞
6:         previous[node] ← undefined
7:     end for
8:     distances[start] ← 0
9:     queue ← PriorityQueue()
10:    queue.insert(start, 0)
11:    while queue is not empty do
12:        current ← queue.pop_min()
13:        if current is goal then
14:            return ReconstructPath(previous, current)
15:        end if
16:        for all neighbor of current do
17:            distance ← distances[current] + cost(current, neighbor)
18:            if distance < distances[neighbor] then
19:                distances[neighbor] ← distance
20:                previous[neighbor] ← current
21:                queue.insert(neighbor, distance)
22:            end if
23:        end for
24:    end while
25:    return Failure
26: end function

27: function ReconstructPath(previous, current)
28:    path ← [current]
29:    while previous[current] is defined do
30:        current ← previous[current]
31:        path.prepend(current)
32:    end while
33:    return path
34: end function
```

The cost function in the A* algorithm typically consists of two components: the actual cost to reach a vertex from the start node, and the heuristic estimate of the remaining cost from the vertex to the goal node. The sum of these two components represents the total estimated cost from the start node to the goal node through a particular vertex.

The heuristic estimate, $h(x_n)$, must be an admissible (underestimating) heuristic function that provides an estimate of the remaining cost from node n to the goal node. Admissibility is a requirement for the algorithm to be able to look for the best solution in the graph. If admissibility is not implemented, overestimation might occur in the algorithm, and it may overlook nodes that would lead to the optimal solution.

The solution for admissible heuristic implementation is based upon the paper from Primatesta et al., 2019. Where the author *dialed down* the heuristic value by multiplying it by the risk cost between x_{goal-1} and the x_{goal} itself, this is to ensure conservative heuristic admissibility.

$$\begin{aligned} h(x_n) = \frac{r(x_n) + r_{\min}}{2} \cdot \operatorname{dist}(x_n, x_{n+1}) + \operatorname{dist}(x_{n+1}, x_{\text{goal}-1}) r_{\min} \\ + \frac{r(x_{\text{goal}}) + r_{\min}}{2} \cdot \operatorname{dist}(x_{\text{goal}-1}, x_{\text{goal}}) \end{aligned} \tag{3.15}$$

By summing the actual cost and the heuristic estimate, the A* algorithm prioritizes nodes with lower total costs, resulting in a search that tends to explore paths that are likely to be closer to the goal. This heuristic-guided exploration helps to efficiently find the shortest path from the start node to the goal node.

To summarize, when there is a single unique optimal path, A* can discover it more efficiently than Dijkstra's algorithm by utilizing an admissible heuristic function. While Dijkstra's algorithm has the capability to eventually find the absolute optimal path given sufficient time. By comparing the results of A* and Dijkstra's algorithm, we can assess the performance of A*.

Algorithm 2 A* Algorithm

```
function AStar(start, goal)
    openSet ← {start}
    closedSet ← {}
    cameFrom ← {}
    gScore ← {}
    hScore ← {}
    fScore ← {}
    gScore[start] ← 0
    hScore[start] ← heuristic(start, goal)
    fScore[start] ← gScore[start] + hScore[start]
    while openSet is not empty do
        current ← node in openSet with the lowest fScore
        if current is goal then
            return ReconstructPath(cameFrom, current)
        end if
        openSet ← openSet \ {current}
        closedSet ← closedSet ∪ {current}
        for all neighbor in Neighbors(current) do
            if neighbor in closedSet then
                continue
            end if
            tentativeGScore ← gScore[current] + cost(current, neighbor)
            if neighbor not in openSet or tentativeGScore < gScore[neighbor]
then
                cameFrom[neighbor] ← current
                gScore[neighbor] ← tentativeGScore
                hScore[neighbor] ← heuristic(neighbor, goal)
                fScore[neighbor] ← gScore[neighbor] + hScore[neighbor]
                if neighbor not in openSet then
                    openSet ← openSet ∪ {neighbor}
                end if
            end if
        end for
    end while
    return Failure
end function

function ReconstructPath(cameFrom, current)
    path ← [current]
    while current in cameFrom do
        current ← cameFrom[current]
        prepend(current, path)
    end while
    return path
end function
```

3.6 UPX, UAV Pathfinding eXtension

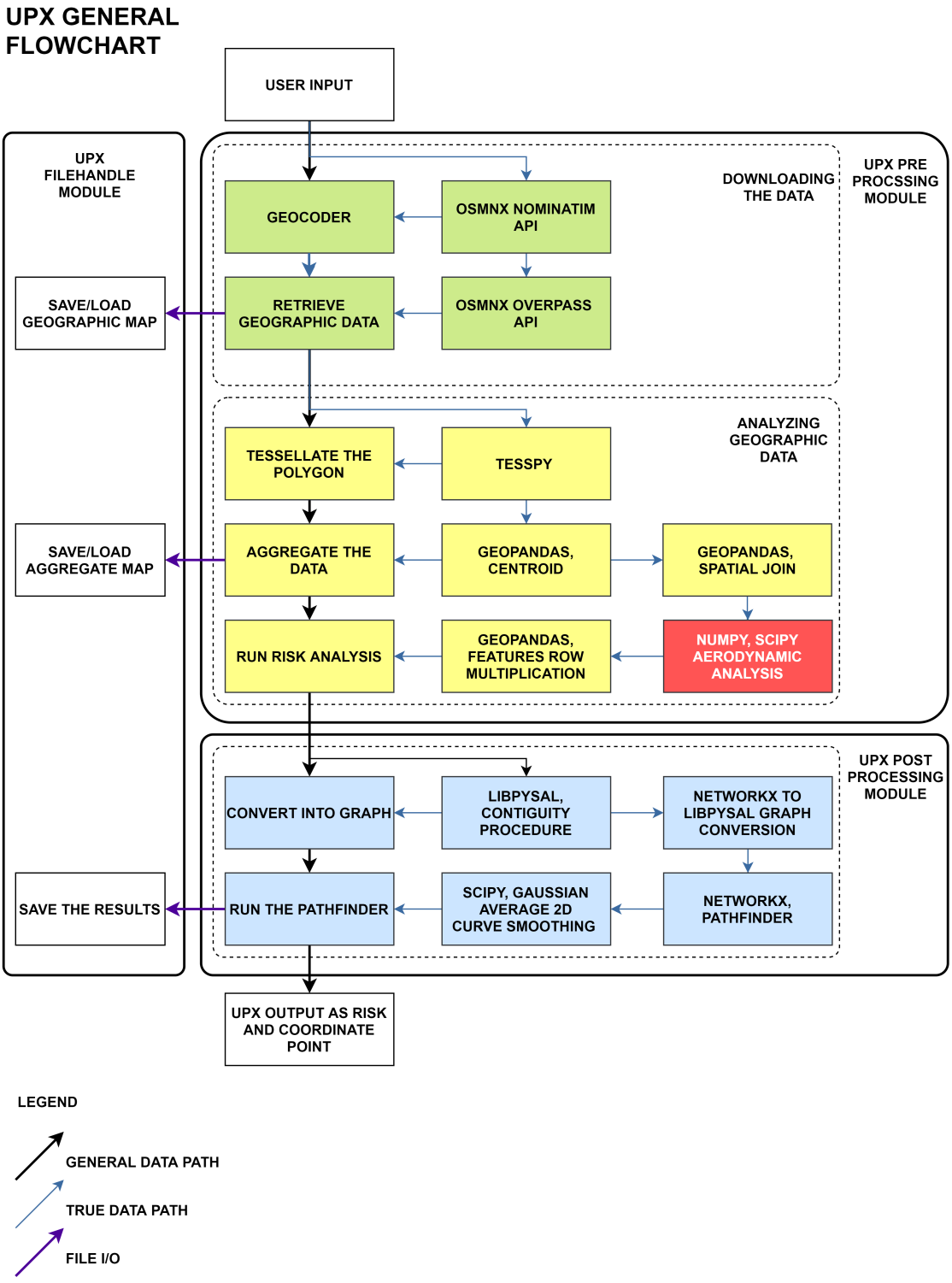


Figure 3.10: UPX architecture design

UAV Pathfinding eXtension, UPX, is both a python libraries and application to find the optimal risk-informed flight path. In general, UPX works by retrieving geographic data from OSM, receiving geographic data, then being discretized and aggregated. The risk map is then converted into a graph for the pathfinding algorithm. The risk mentioned Risk in UPX is rules to determine the probability of fatality in case the UAV happens to malfunction and falls in that area.

UPX was built based on the concern of negative consequences of increase frequency of UAV flying in urban areas. While UPX is intended to be used to aid UAV operation, its library can be used in any type of aircraft as long they provide aerodynamic parameters and aircraft performance data. However, some caveats would be explained in the UPX limitation below.

3.6.1 Limitation

Firstly, UPX is strictly for a preliminary assessment only. It does not handle real-world phenomena such as sudden wind gale, rain, storm, or other aircraft that, unfortunately, are present in the path of the UAV. For that reason only, we hope the pilot practice safe flying and the UAV itself is provided with the failsafe feature.

3.7 UPX Python and packages

3.7.1 Python

Python is an interpreted programming language. Interpreted language has the benefit of being an easier language to compare to a compiled language. Added to this, it has massive scientific-related packages and libraries. Combine with quick deployment and its STEM library, it is a perfect choice for engineering, mathematics, and physics analysis. Based on previous criteria it is a great choice for creating UAV path planner applications and libraries.

While Python indeed has a great advantage. Unfortunately, its comparatively easy language came with a great price. An interpreted language is inherently slow, 8 to 100 times slower compared to C/C++ language. Base Python language also cannot use more than one thread because its GIL, Global Interpreter Lock, prevents the threads from sharing memories and create a parallel workload. Fortunately, most of the heavy calculation is handled by the c/c++ library, where Python uses the caller function to launch the libraries.

3.7.2 GeoPandas

GeoPandas is one of the GIS libraries built for tabulating and analyzing geographicalspatial analysis in python. GeoPandas itself can be regarded as the extension of Pandas library, where Pandas is general tabular data processor. GeoPandas has geographic spatial and attribute related function such as overlay, spatial join, clipping to group, aggregate, and measure geospatial data.

GeoPandas exports and imports files in standard GIS-supported files such as geoJSON, the file, and the Shapely object, to name a few. In Python GeoPandas tabular data is called GeoDataFrame, and GeoSeries. What differentiates between GeoDataFrame and regular DataFrame is the addition of a geometry column. The geometry column can contain Multi/Polygon, Multi/Linestring, and Point shapes.

3.7.3 Shapely

Shapely is a geomeytry calculation and manipulation packages available for python. Many of geospatial function in GeoPandas relies on Shapely itself. Overlay, clipping, spatial binary combination in GeoPandas can be found in Shapely. In this paper Shapely is being uses to create custom filter for heat map function of aircraft falling probability.

3.7.4 Tesspy

Tesspy, a tesselation library for Python. Its purpose, as the name implies, is to create a tesselated polygon. Tesspy can be used as long as the polygon object is derived from the Shapely object. Tesspy offers type of tessellated map that commonly use for geospatial processing. Square grid, hexbin, Voronoi city block, and adaptive square is the type of tesselated maps, to name a few.

3.7.5 OSMnx

OSMNx is a combination of OSM and NetworkX. OSMNx is mainly being uses to create graph representation of the road network. It uses the GeoPandas data frame, to build the network, as long as the roads in question are connected at least with another road. But for this paper, OSMNx is only being used as a geocoder and reverse geocoder to retrieve GeoPandas data.

3.7.6 NetworkX

To create a graph structure, NetworkX provides the necessary function to do that. NetworkX offers graph creation, graph traversal algorithms, and graph manipulation. The graph, as previously mentioned, in a combination of nodes and edges. The A* pathfinder for the UAV Path finder also uses the NetworkX built-in A*.

CHAPTER 4

RESULTS

4.1 Ground risk map generation

4.1.1 UAV Specification

In this study, we are going to analyze the risk factor of multirotor and fixed-wing UAV. In comparison, Talon and Parrot Disco are fixed-wing. While DJI Phantom 4 and DJI Inspire 2 are multirotor. Four of them are chosen since they are generally known as a go-to standard in the commercial drone industry. Each category is also differentiated into two different sizes, where Talon and Inspire 2 are the large size, while Phantom 4 and Disco are much smaller UAV. All properties are obtained from manufacturer design specifications.

As for the aerodynamic properties of these drones, we are using a very simple drag model. This will not include an Induced drag since we cannot predict every possible angle of attack. It is also beneficial to use simplify drag model since it would give a higher impact velocity, which increases the impact energy and results in a much more conservative value.

It is worth mentioning the mechanic of the glide model in this risk analyzer. Since only fixed wings are capable of gliding, and to an extent, a variable-pitch helicopter by autorotation, it is almost impossible to make a glide descent on a fixed-pitch propeller multirotor. Where for the multirotor, ballistic descent is the more likely outcome, even though there is little to no damage in the aircraft. It is then assumed that a glide descent, and to an extent a glide speed in multirotor, as a low powered horizontal descent where the power is barely sustaining a hover maneuver

Specification	**Talon**	**Parrot Disco**	**Phantom 4**	**Inspire 2**
Type	Fixed wing	Fixed wing	Quadcopter	Quadcopter
Mass (kg)	3.75	0.75	1.4	4.25
Front area (m^2)	0.1	0.07	0.02	0.04
Radius (m)	0.88	0.575	0.2	0.4
Maximum flight time	1.25 h	45 min	20 min	25 min
Ballistic drag coefficient	N(0.9, 0.2)	N(0.9, 0.2)	N(0.7, 0.2)	N(0.7, 0.2)
Horizontal initial speed (m/s)	N(18, 2.5)	N(15, 2.5)	U(0, 15)	U(0, 20)
Vertical initial speed (m/s)	0	0	0	0
Glide speed (m/s)	16	12	7.5	10
Glide ratio	N(12, 2)	N(2.7, 0.8)	N(2.7, 0.8)	N(2.7, 0.8)
Parachute drag coefficient	N(13, 0.2)	N(0.9, 0.2)	N(0.9, 0.2)	N(0.9, 0.2)
Parachute area	1	0.5	0.5	0.5

Table 4.1: UAV Specifications

4.2 Objective

The ground risk would be performed in North Jakarta, Indonesia. The place is chosen for its dense population, an assortment of building heights, large water bodies, and multiple residential and commercial zone areas. Shown in 4.1

And for the operation itself, the UAV would be flying from Pluit Village to Jakarta Sailing Vocational School. This objective is chosen based on the criteria previously mentioned.

We would compare the performance of the Risk A* Algorithm vs the Normal A* algorithm. We do not include Djikstra's algorithm in this section since the A* algorithm provided by the NetworkX library is already admissible and able to obtain optimal pathways.

It is worth mentioning the normal A* is the same for all types of aircraft since aircraft properties are only important in Risk A* in generating various descent PDFs.

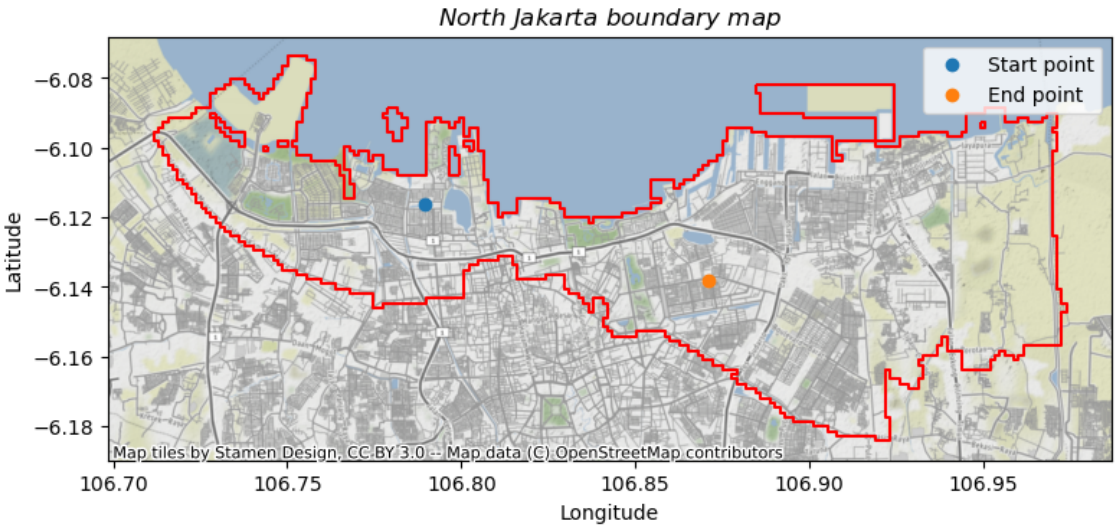


Figure 4.1: Operation objective

4.3 PDF risk map

4.3.1 Talon

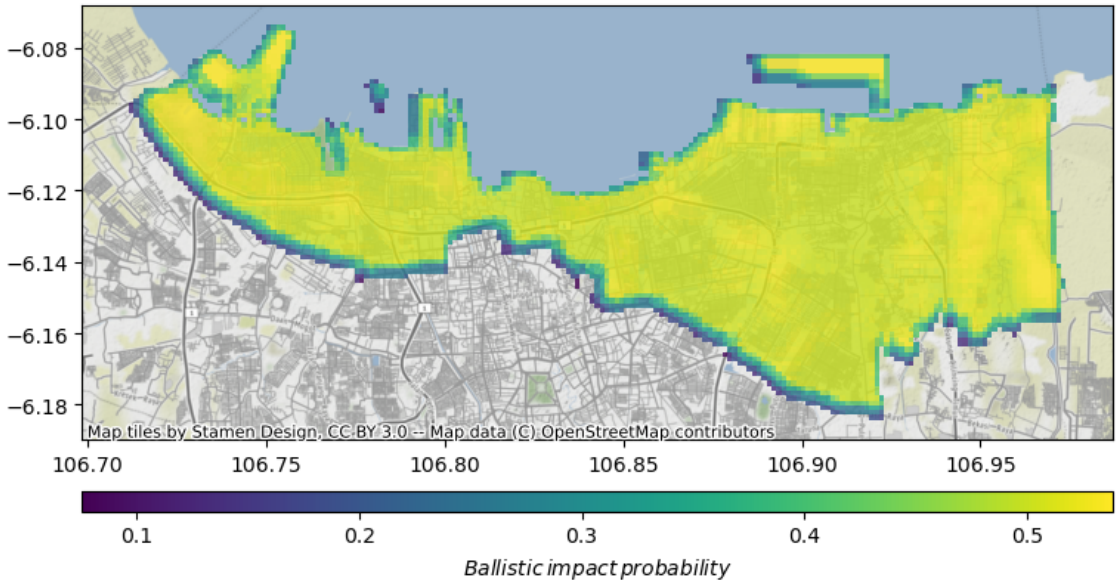


Figure 4.2: Talon ballistic PDF

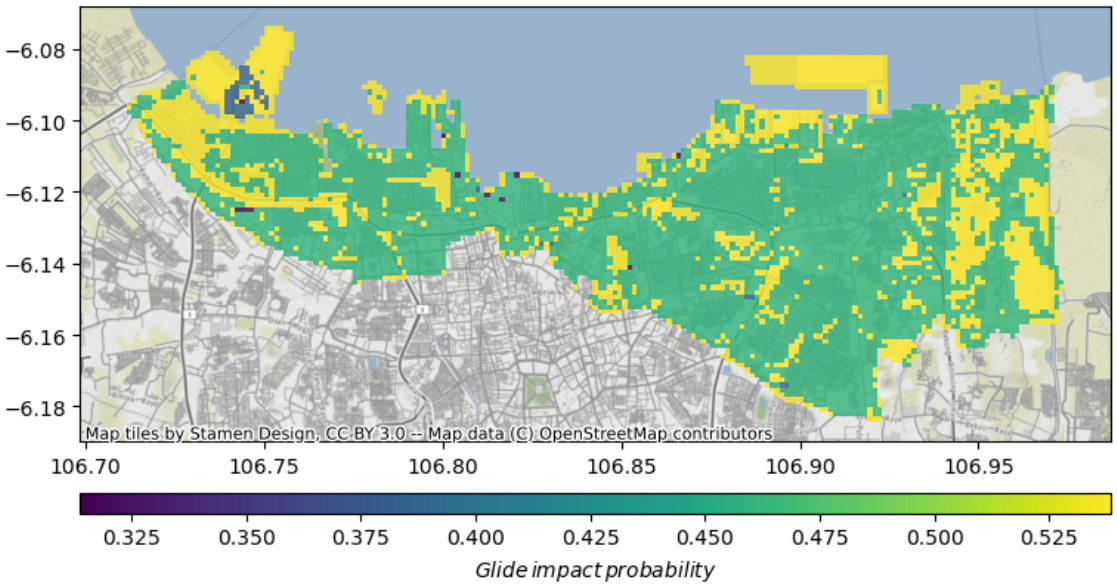


Figure 4.3: Talon glide PDF

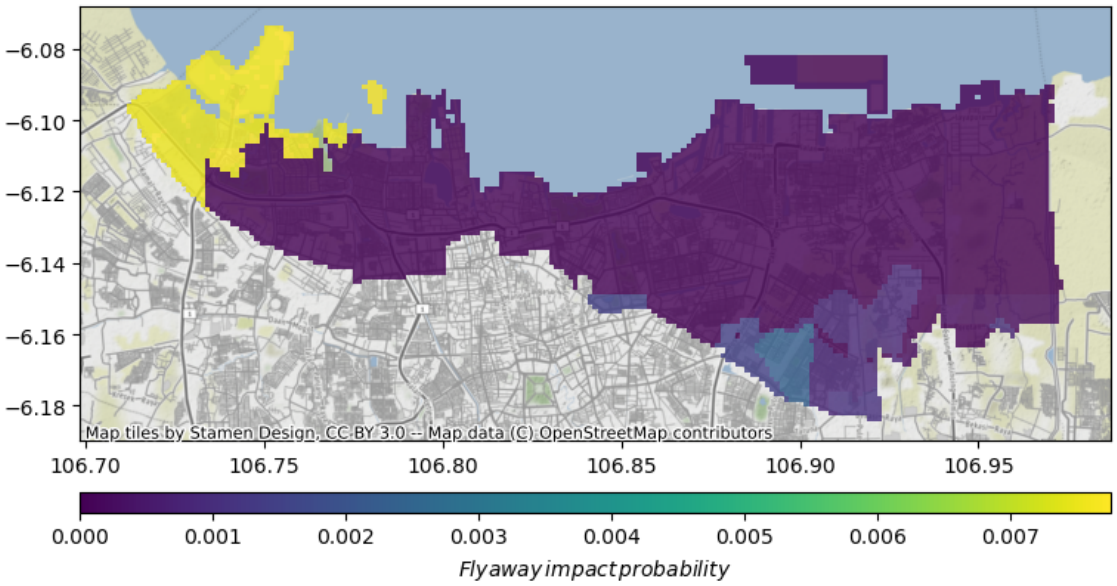


Figure 4.4: Talon fly away PDF

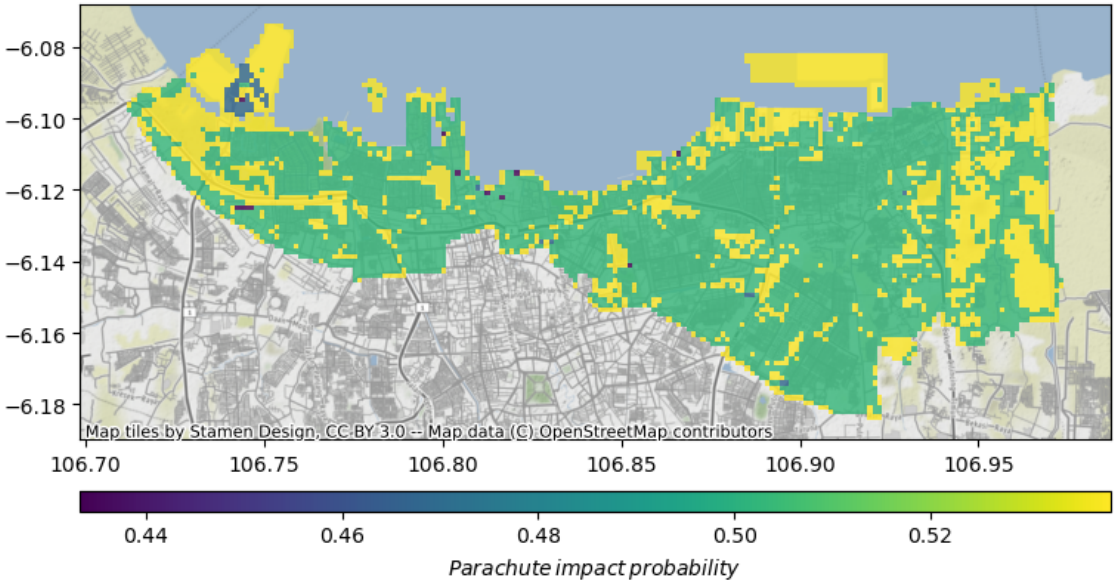


Figure 4.5: Talon parachute PDF

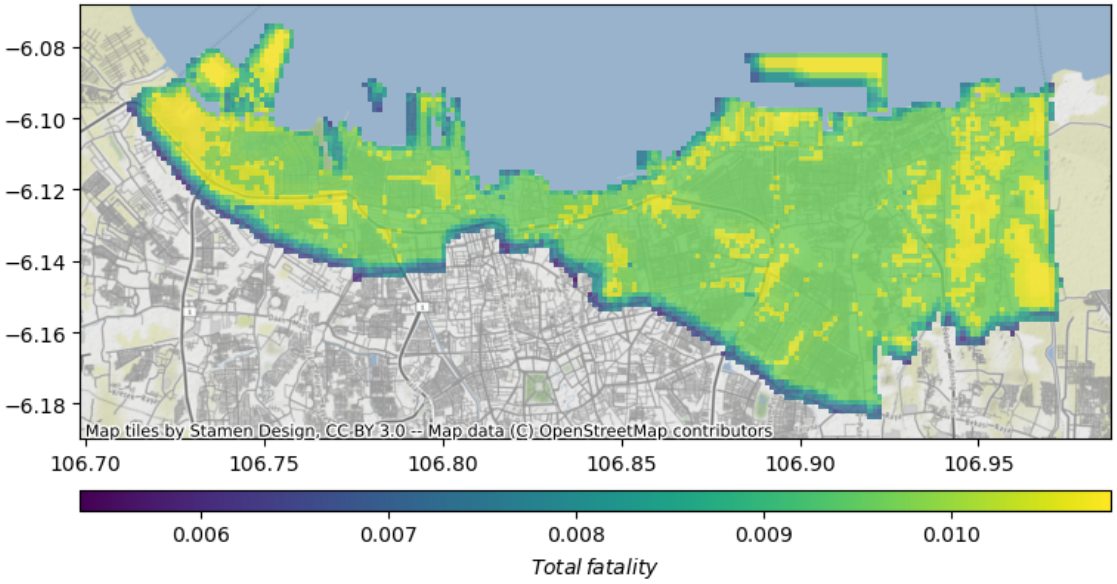


Figure 4.6: Talon fatality risk PDF

4.3.2 Parrot Disco

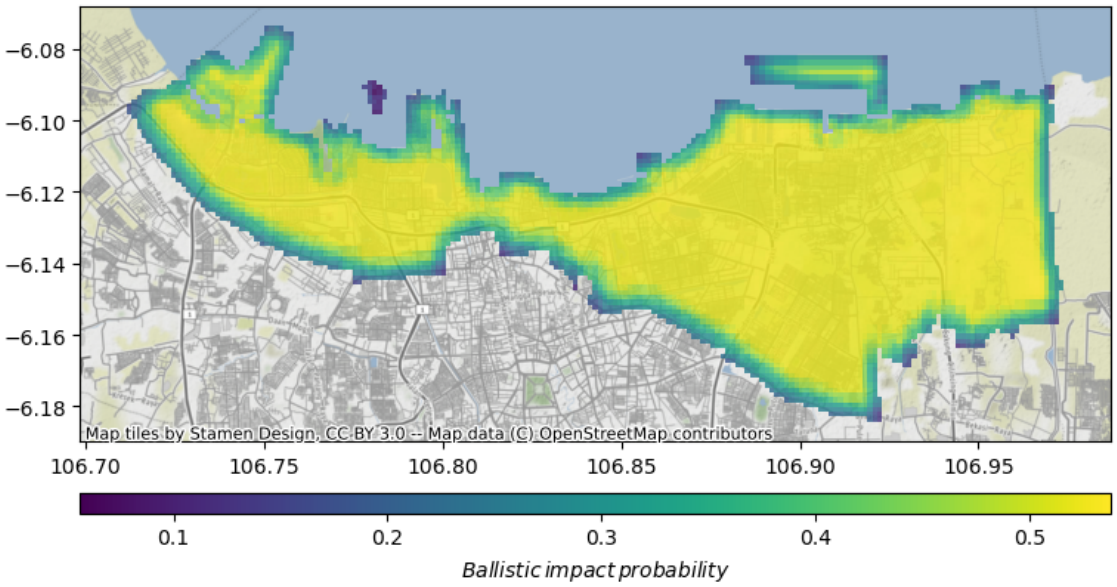


Figure 4.7: Disco ballistic PDF

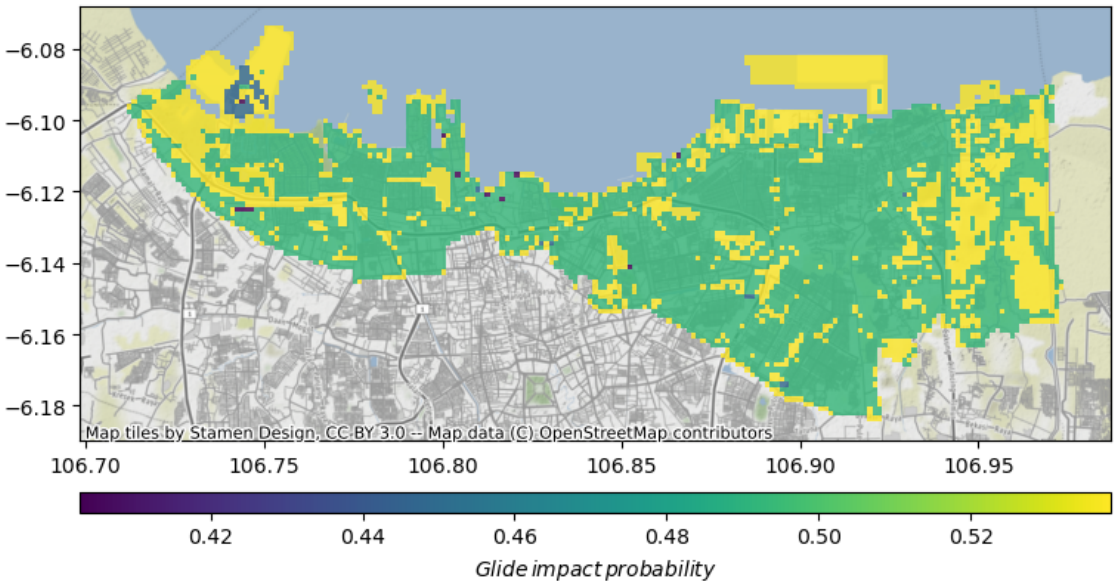


Figure 4.8: Disco glide PDF

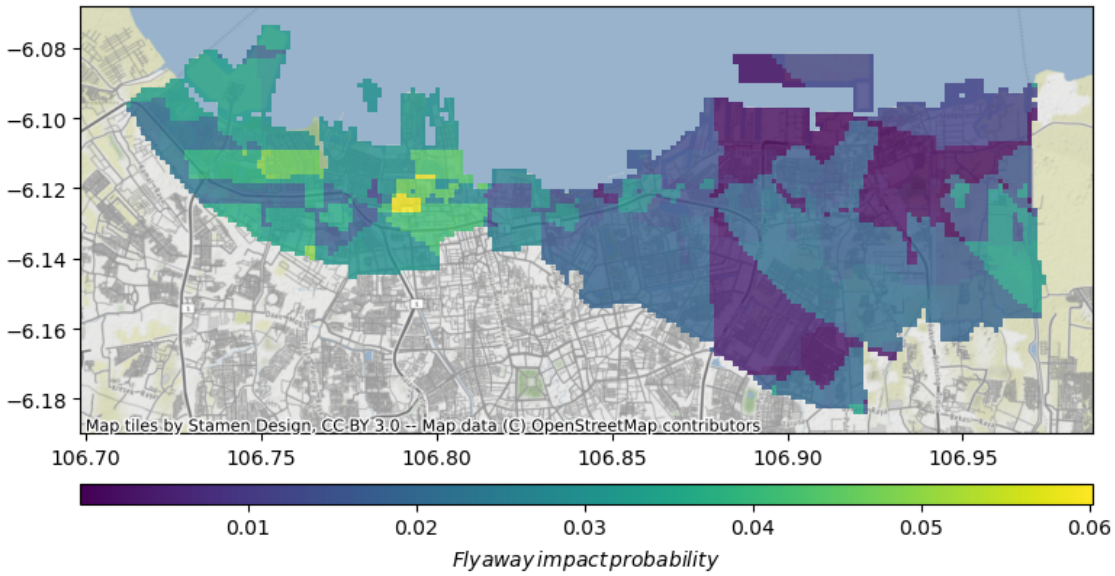


Figure 4.9: Disco fly away PDF

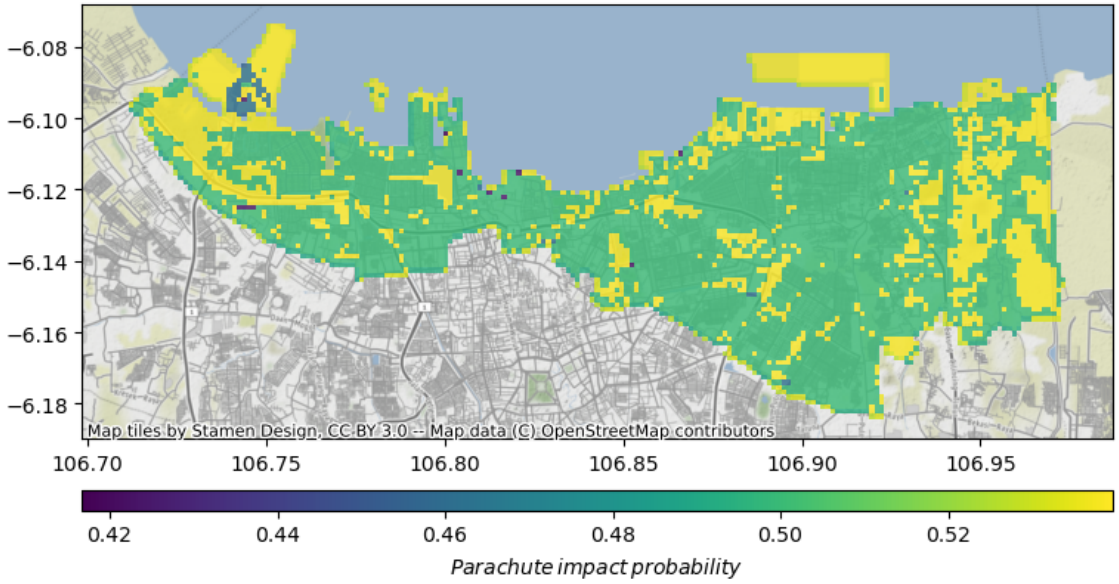


Figure 4.10: Disco parachute PDF

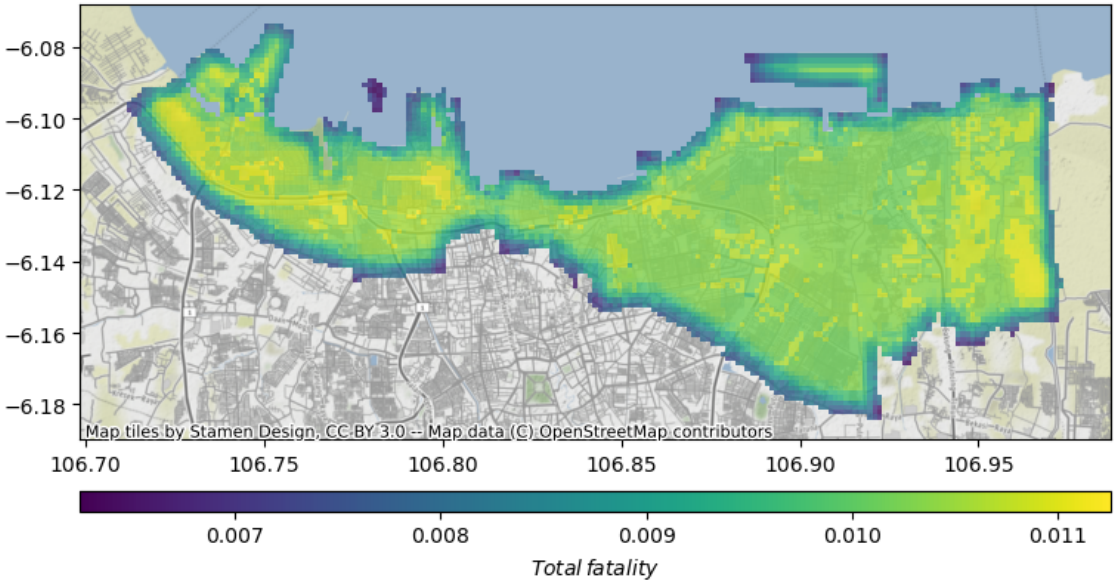


Figure 4.11: Disco fatality risk PDF

4.3.3 DJI Phantom 4

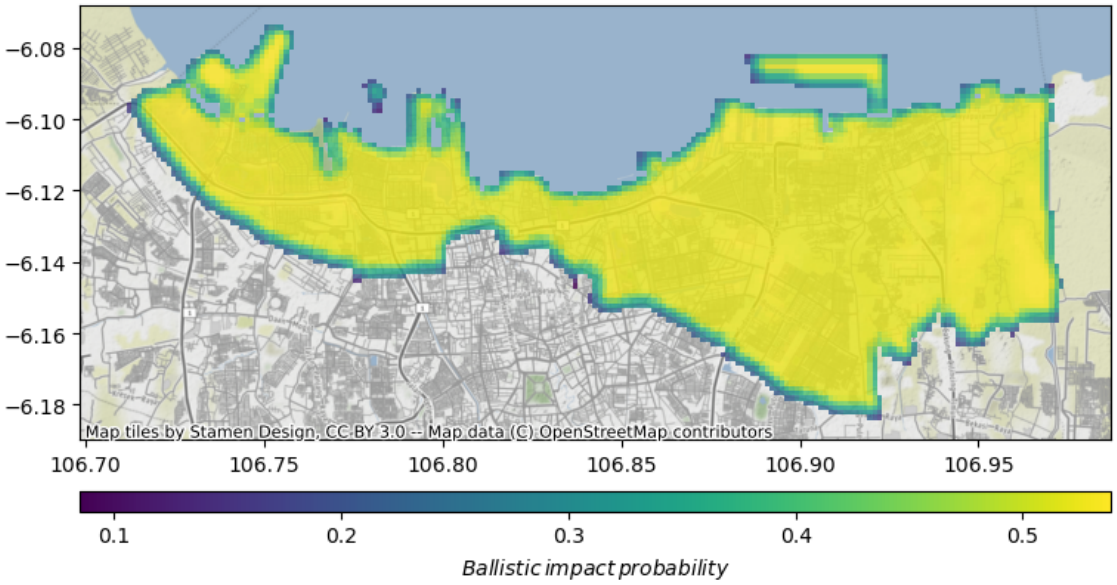


Figure 4.12: DJI Phantom 4 ballistic PDF

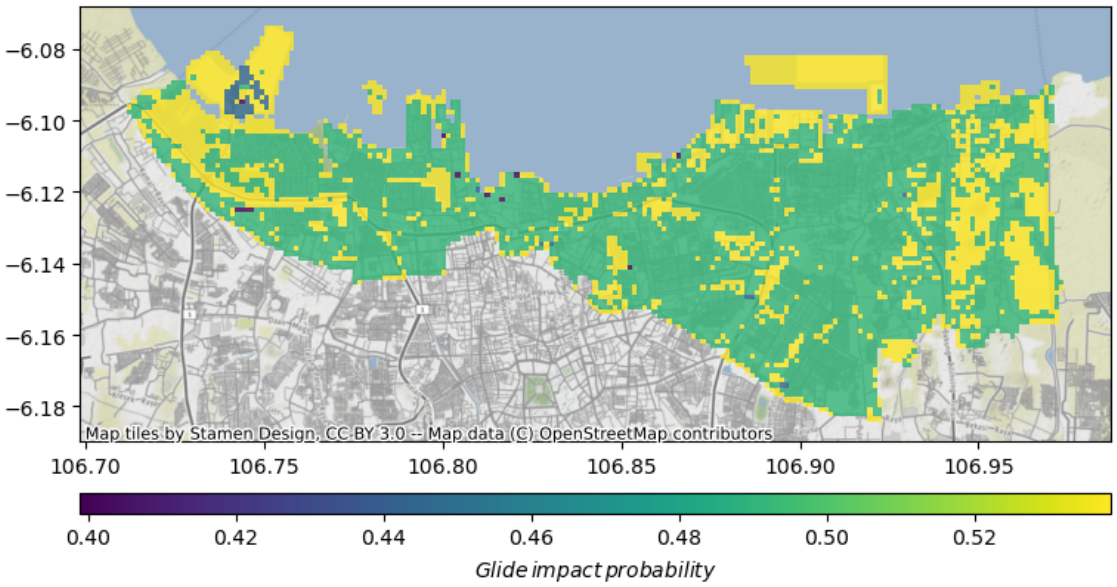


Figure 4.13: DJI Phantom 4 glide PDF

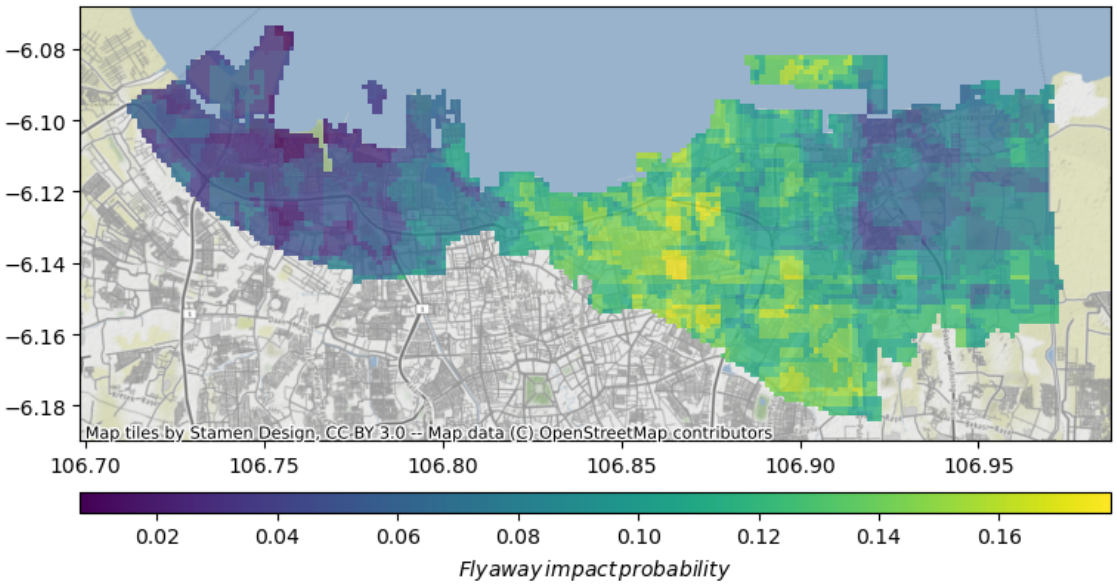


Figure 4.14: DJI Phantom 4 fly away PDF

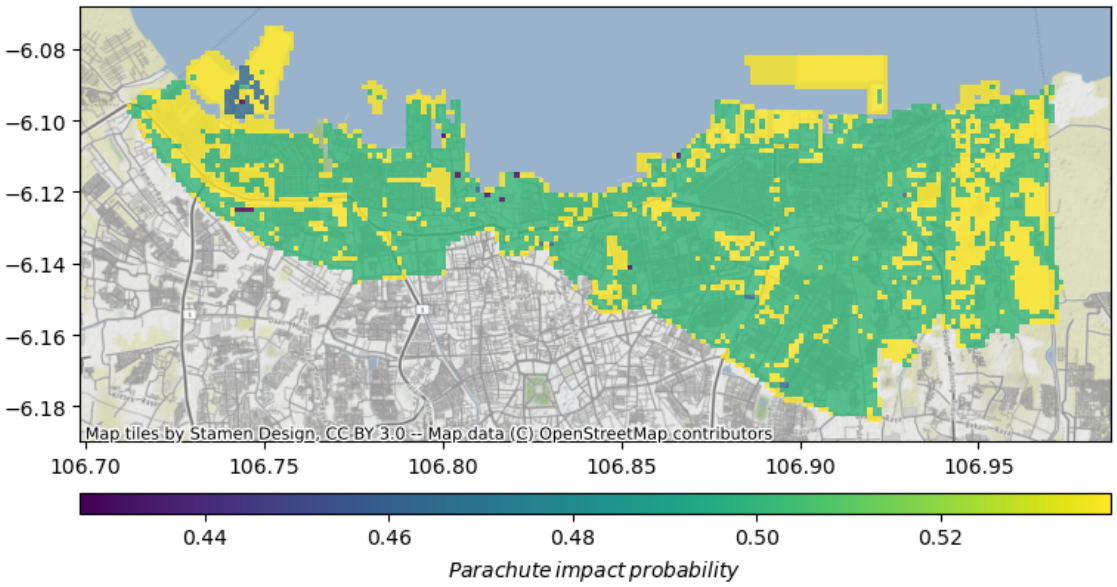


Figure 4.15: DJI Phantom 4 parachute PDF

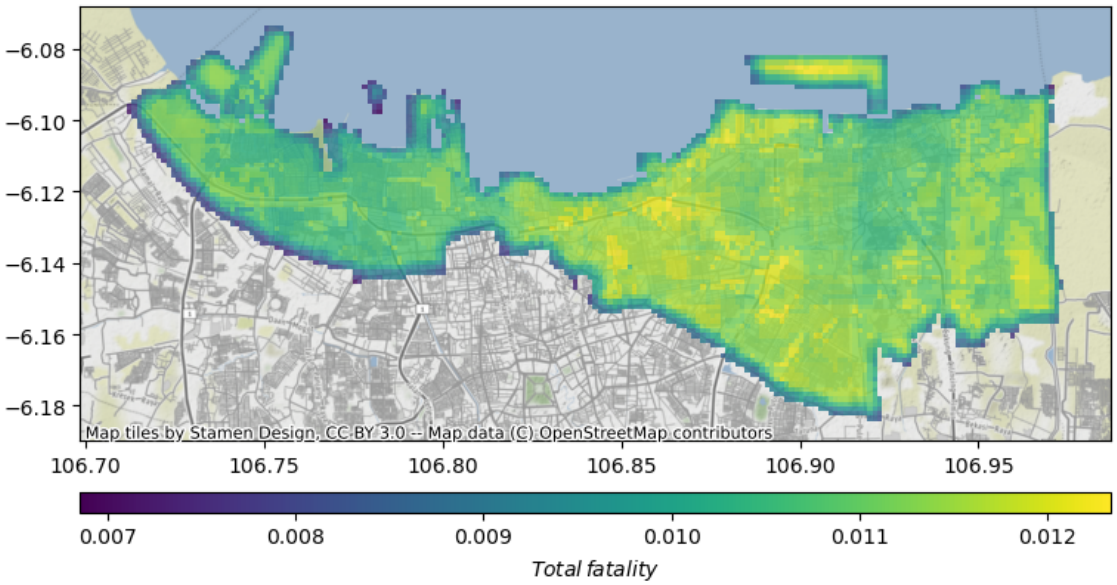


Figure 4.16: DJI Phantom 4 fatality risk PDF

4.3.4 DJI Inspire 2

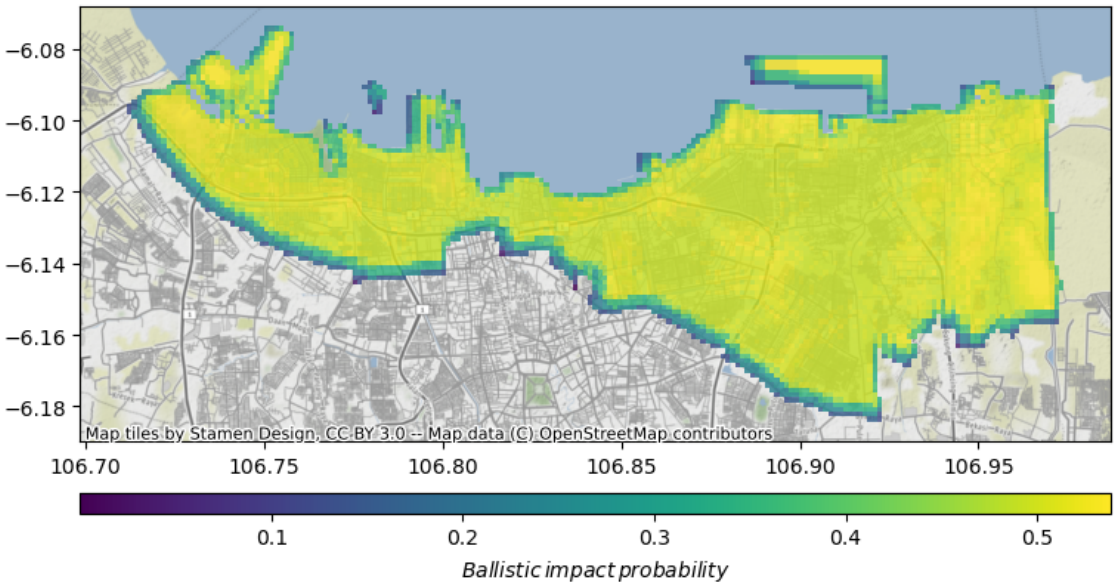


Figure 4.17: DJI Inspire 2 ballistic PDF

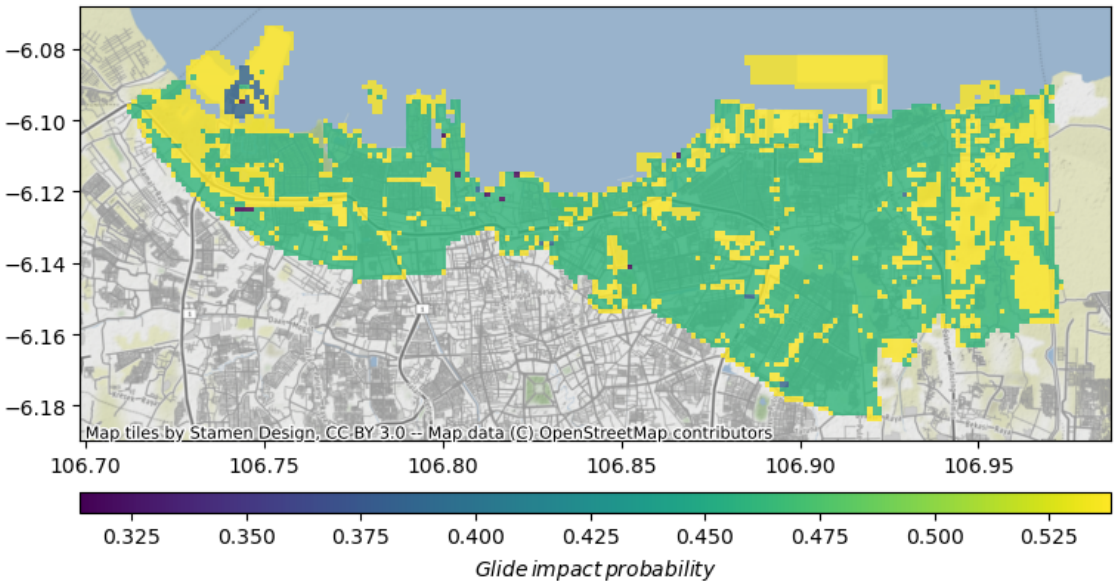


Figure 4.18: DJI Inspire 2 glide PDF

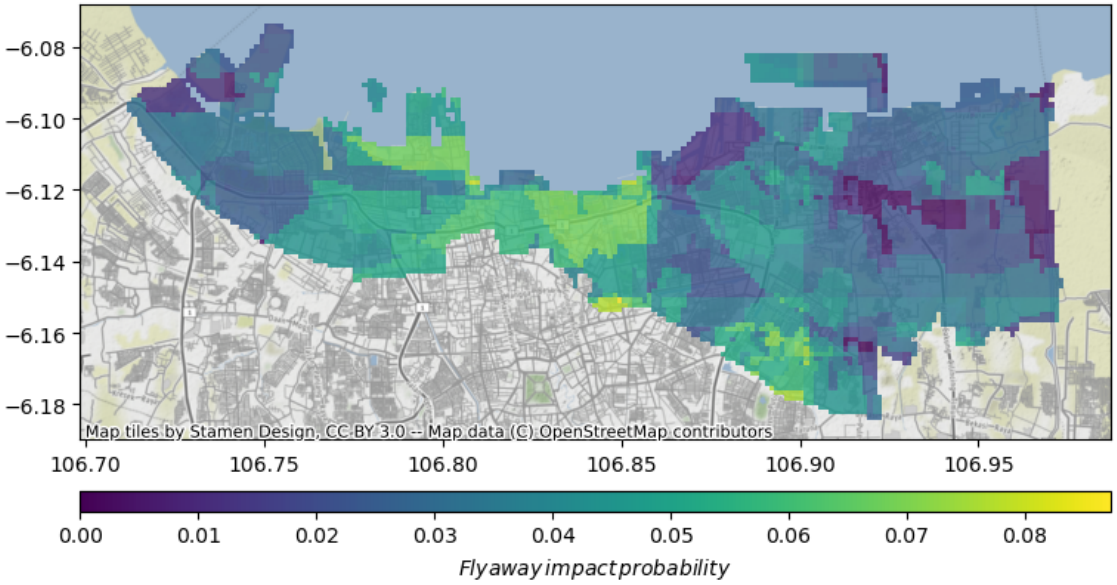


Figure 4.19: DJI Inspire 2 fly away PDF

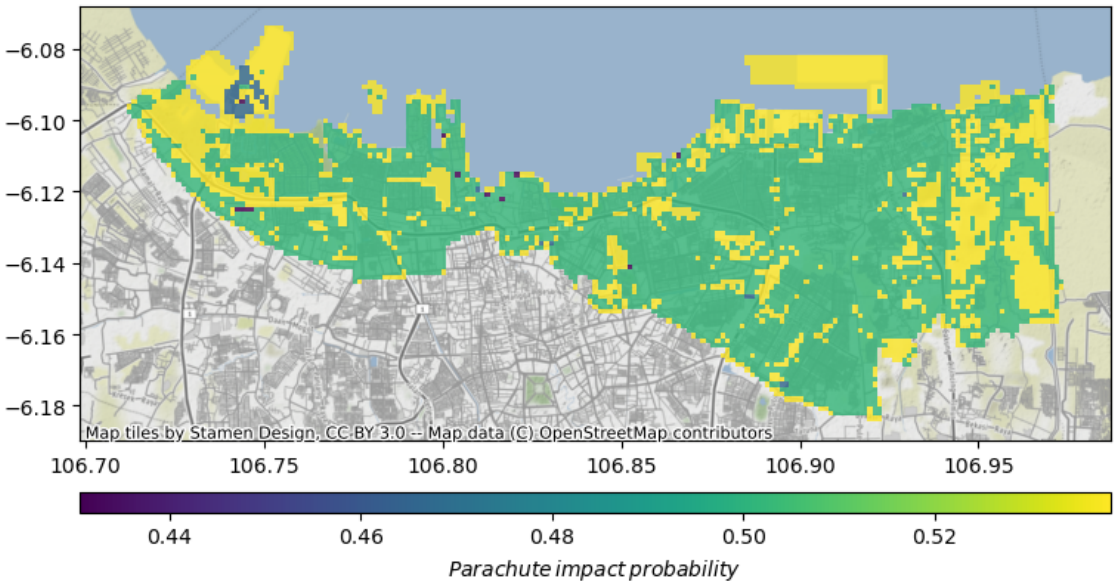


Figure 4.20: DJI Inspire 2 parachute PDF

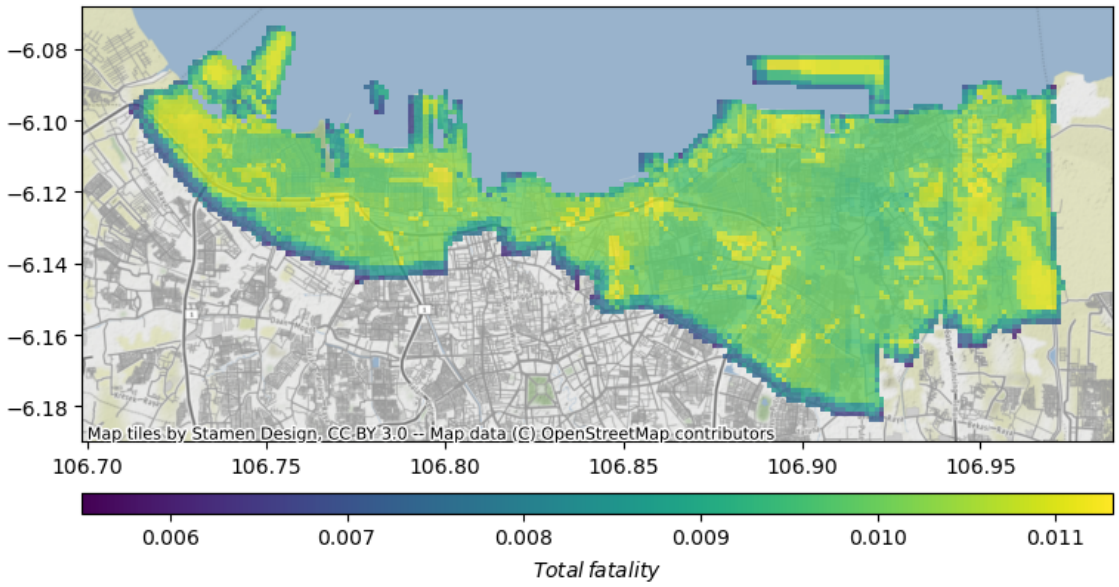


Figure 4.21: DJI Inspire 2 fatality risk PDF

4.4 Pathfinder result

4.4.1 Regular A*

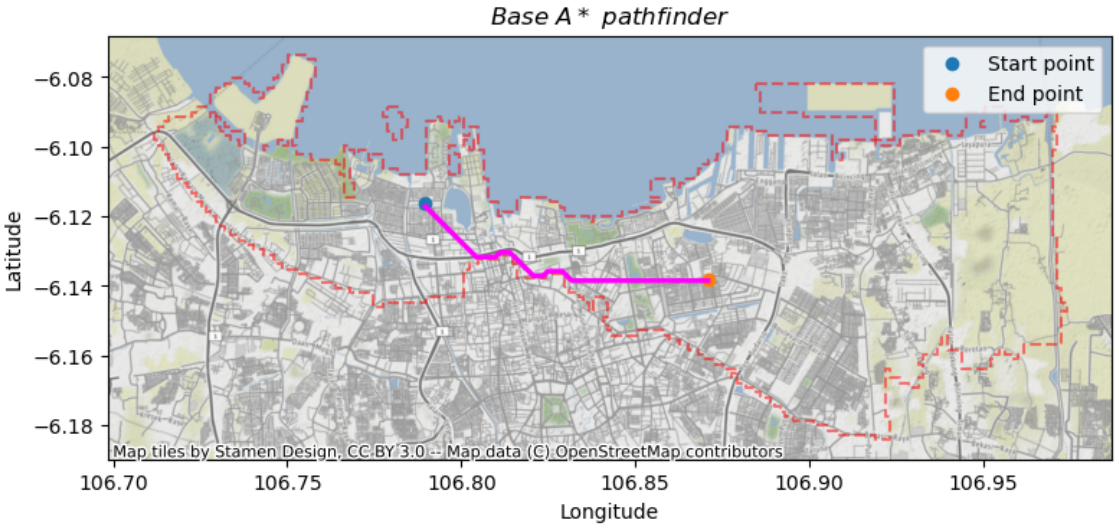


Figure 4.22: Regular A* flight path

4.4.2 Risk A*

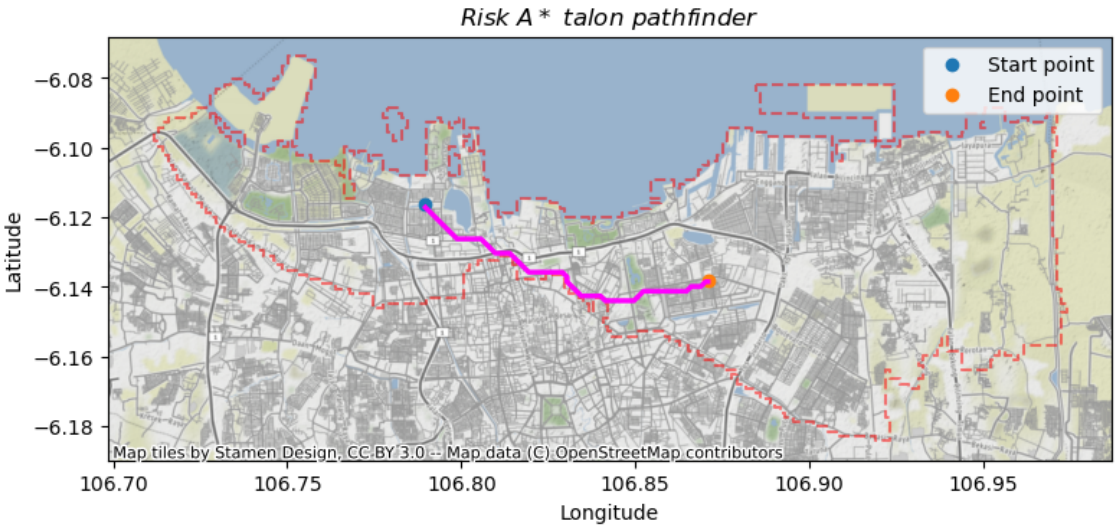


Figure 4.23: Talon risk A* flight path

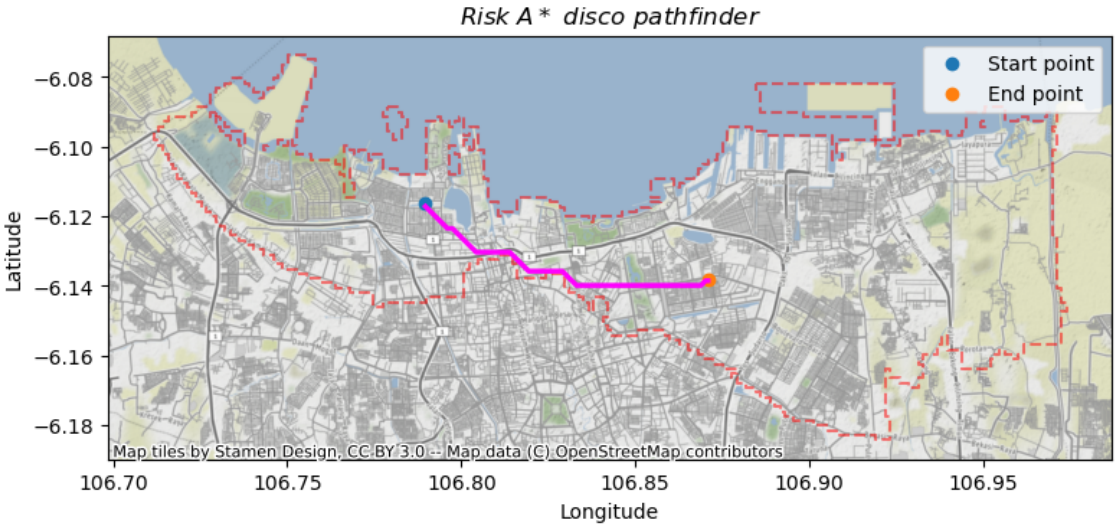


Figure 4.24: Disco risk A* flight path

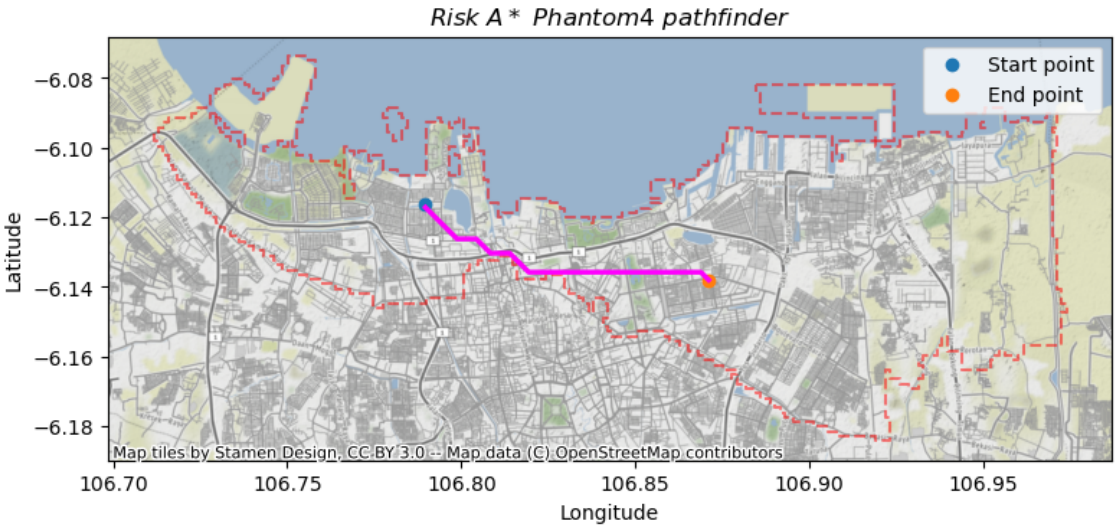


Figure 4.25: DJI Phantom 4 risk A* flight path

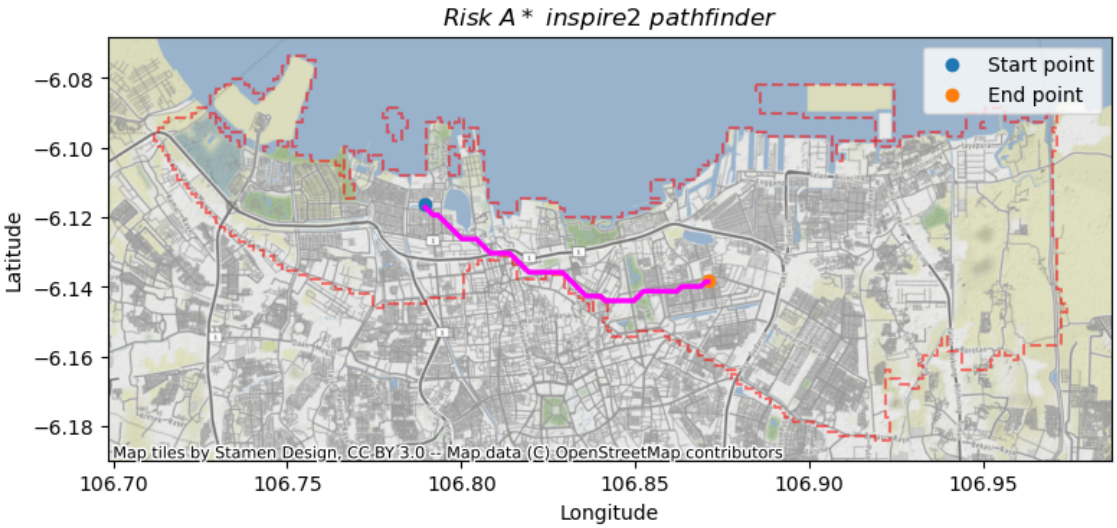


Figure 4.26: DJI Inspire 2 risk A* flight path

4.4.3 Risk result

From 4.4, It is not surprising that the traveled distance of Normal A* is the shortest of the rest of the risk A*. And also, it is interesting to see that the normal A* has the highest risk, although it is logical to think that normal A* inversely correlates to its risk A* counterpart. It is not guaranteed for a normal A* to have the highest risk score. However, normal A* will guarantee to have the shortest path length compared to risk A*.

To look at different properties, we can also measure the risk-to-traveled distance ratio, which can be thought of as how much the algorithm tried to avoid critical areas by extending the pathway to a much safer option. This example is clearly

shown in DJI Phantom 4, where DJI Phantom 4 shares almost the same risk as normal A* with a 0.02 difference, But takes a pathway that is 163 meters longer.

After looking at the graph, it is tempting to use Talon since it has the lowest traveled distance and also risk score. However, this resulted from the limitation in the way that the program is built, which of course, needs some improvement.

The problem lies in the Flyaway descent of this aircraft. This is because Talon has a long-range flight before crashing into the ground. Now the main problem is since we only use 1 cluster of geographic regions, North Jakarta, the region is smaller than the flyaway radius. This resulted in the outer cells of the region having a smaller risk compared to the center cells.

It is then clear the score represented by shorter-range UAV such as Parrot Disco, DJI Phantom 4, and DJI Inspire 2 has a more defined score compared to the score produced by Talon.

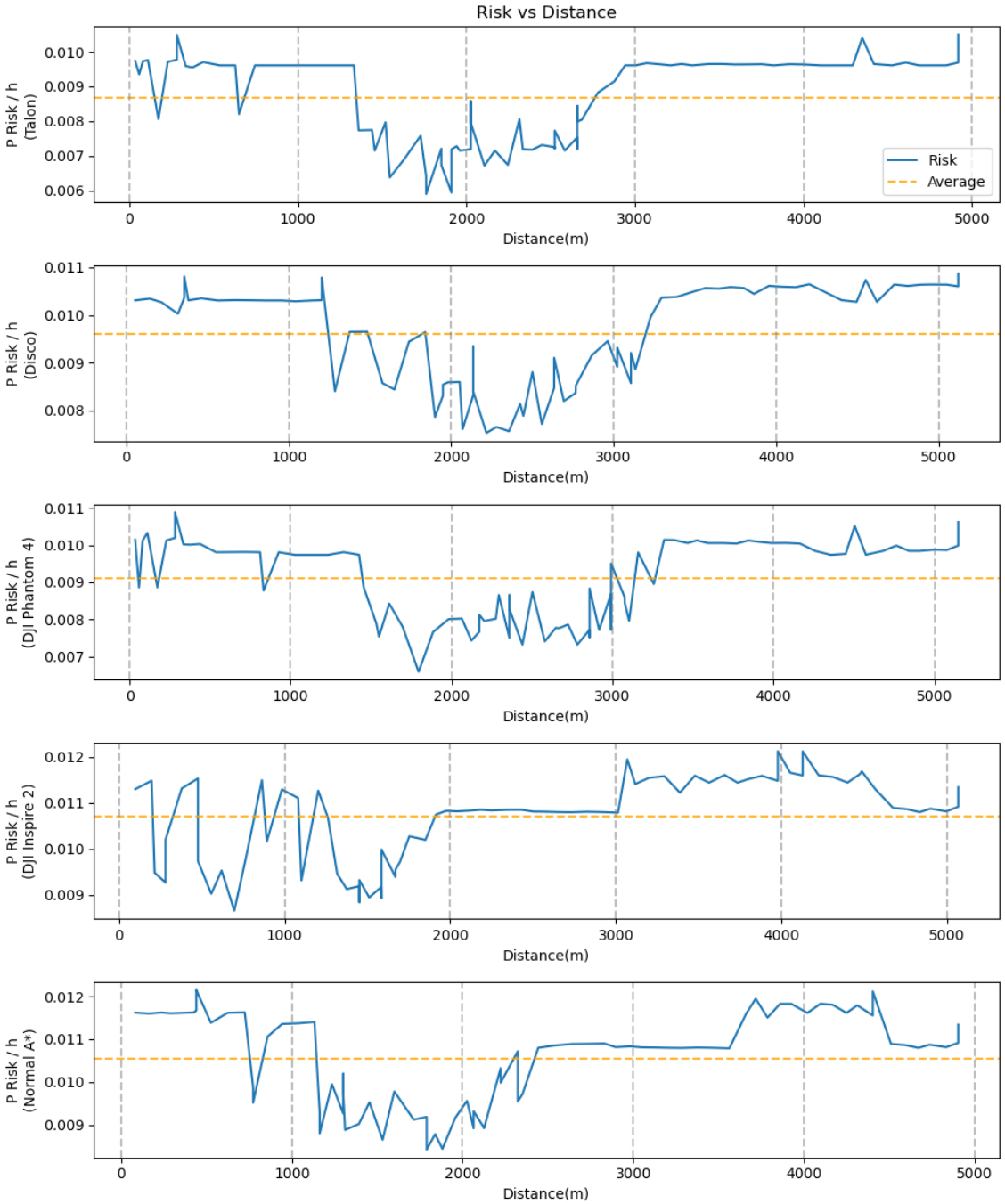


Figure 4.27: Aircraft risk score

Parameter	**Talon**	**Disco**	**Phantom 4**	**Inspire 2**
Total travelled distance	4917.651992	5120.845475	5066.796306	5147.428533
Total risk involved	0.727661	0.759534	0.834316	0.765401
Risk vs distance ratio	0.000148	0.000148	0.000165	0.000149

Table 4.2: Total pathway risk (%)

Drone	Stat	Ballistic	Glide	Flyaway	Final Risk
Talon	min	7.55e-02	3.14e-01	0.00e+00	5.36e-03
	max	5.38e-01	5.38e-01	7.76e-03	1.08e-02
	avg	4.74e-01	4.83e-01	8.47e-04	9.57e-03
Disco	min	5.63e-02	4.03e-01	5.38e-05	6.25e-03
	max	5.38e-01	5.38e-01	6.02e-02	1.13e-02
	avg	4.66e-01	5.06e-01	1.93e-02	9.91e-03
DJI Phantom 4	min	8.50e-02	3.99e-01	7.19e-03	6.85e-03
	max	5.39e-01	5.38e-01	1.79e-01	1.23e-02
	avg	4.87e-01	5.05e-01	9.19e-02	1.08e-02
DJI Inspire 2	min	6.03e-05	3.14e-01	0.00e+00	5.50e-03
	max	5.38e-01	5.38e-01	8.72e-02	1.13e-02
	avg	4.63e-01	4.83e-01	3.50e-02	9.81e-03

Table 4.3: Pathway risk statistic (%)

	Talon	Disco	Phantom 4	Inspire 2
Ballistic impact probability	3357.142553	3303.336594	3452.695201	3284.220663
Fly away impact probability	6.004905	136.521018	651.025645	248.005463
Glide impact probability	3420.454716	3584.184491	3576.983890	3420.454716
Parachute impact probability	3635.634721	3616.844608	3624.644006	3631.138685
Total fatality risk probability	67.836022	69.526808	76.807047	69.526808
Average areal total fatality risk	0.009572	0.009810	0.010838	0.009810

Table 4.4: Combine total map risk statistic (%)

CHAPTER 5
SUMMARY, CONCLUSION, AND RECOMMENDATION

5.1 Summary

In the summary ground, a risk map is a discrete PDF in the shape of a grid containing the possible fatality risk of each cell element. It is obtained by multiplying the possible outcome of multiple layers contributing to each risk. These layers are population density, no-fly zone layer, obstacle layer, shelter layer, and descent layer.

- The population layer is obtained by measuring the number of people per area of the cell.
- The no-fly zone layer represents military and aerodrome space and its critical infrastructure that the UAV must not fly into
- The obstacle layer represents a layer where the height of the geographic features becomes an obstacle depending on the flight level.
- The shelter factor represents the capability of the building to absorb the impact energy of a crashing UAV.
- The descent layer is a layer that describes the probability of the drone impacting the area. The descent layer is split into 3 parts that additively contribute to the descent layer. These parts are ballistic descent, glide descent, parachute, and flyaway.

After building the ground risk map. the next step would be finding the optimal Risk A* flight path where Risk A* is a modified A* pathfinding algorithm, which uses risk and distance to measure the most optimal path.

5.2 Conclusion

Using ground risk management to measure the total area fatality of impact risk resulted in an average of:

- Talon having 9.57×10^{-3} fatality per flight hour
- Parrot Disco having 9.91×10^{-3} fatality per flight hour
- Phantom 4 having 1.08×10^{-2} fatality per flight hour
- Inspire 2 having 9.81×10^{-3} fatality per flight hour

With a traveled distance of each aircraft being:

- Talon, with 4,918 m
- Parrot Disco, with 5,121 m
- Phantom 4, with 5,066 m
- Inspire 2, with 5,147 m
- Any Aircraft Without Risk involved, with 4,903 m

And average path risk of aircraft:

- Talon, with 1.48×10^{-4} fatality per flight hour
- Parrot Disco, with 1.48×10^{-4} fatality per flight hour
- Phantom 4, with 1.65×10^{-4} fatality per flight hour
- Inspire 2, with 1.49×10^{-4} fatality per flight hour
- Any Aircraft Without Risk involved, with 1.74×10^{-4} fatality per flight hour

5.3 Recommendation

5.3.1 For future research

Create an improvement in the methodology to use a much more elegant solution to smooth out the curve by using the combination of the Dublin curve and spline interpolation instead of using a simple spline method. Create a feedback loop where the risk re-apply itself to the ground risk pdf for each time step to give a much more realistic scenario.

Use a space partitioning algorithm such as a binary tree or RTree for the geographic area. This is to prevent over-boundary risk problems that Talon encountered. This is because the binary tree will expand to every region that overlaps with the furthest flight distance. Not only that, the partitioning algorithm can theoretically increase the performance of calculation since it frees up memory when the partitioned area is not being calculated.

5.3.2 For software design

And as for the software design, UPX is currently written purely inPython. This is because it is for proof of concept for the research paper, where the development time is much more critical than the performance. However, if in the future there are similar methods and come across in using UPX. These are some key notes to improving the software.

In order to effectively compute the risk map in great detail for each region, it is highly recommended to utilize distributed systems such as Apache Spark and Dask. These platforms are designed to handle massive amounts of data efficiently by distributing the workload across multiple nodes, allowing for parallel processing and significant reductions in computation time.

Apache Spark is a powerful distributed computing framework that can process large datasets quickly through in-memory computing capabilities. It supports a wide range of operations, including SQL queries, streaming data, and complex analytics. By leveraging Spark's robust ecosystem, we can perform extensive risk calculations and data analyses across large geographic areas without the bottlenecks associated with traditional single-machine processing, The Apache Software Foundation, 2024.

Dask is another excellent option for distributed computing, particularly for Python-based workflows. It enables parallel computing by extending Python's native data structures, such as NumPy arrays and Pandas DataFrames, to operate in a distributed manner. Dask is highly flexible and can seamlessly integrate with existing Python code, making it an ideal choice for scaling up data processing tasks and risk map computations, Dask Development Team, 2016.

Implementing these distributed systems offers several advantages:

- **Scalability:** Both Spark and Dask can scale horizontally by adding more nodes to the cluster, accommodating growing data volumes and computational demands.
- **Efficiency:** Distributed computing significantly reduces the time required to process large datasets, enabling faster and more detailed risk assessments.
- **Flexibility:** These platforms support a wide range of data processing and analysis tasks, making them versatile tools for comprehensive risk mapping and other analytical needs.
- **Resilience:** Distributed systems are designed to handle node failures gracefully, ensuring reliable and continuous operation even in the face of hardware or software issues.

By integrating Spark or Dask into the software design for risk map computation, we can achieve a higher level of detail and accuracy in risk assessments, ultimately leading to more informed decision-making and better management of potential hazards.

BIBLIOGRAPHY

Aalmoes, R., Cheung, Y., Sunil, E., Hoekstra, J., & Bussink, F. (2015). A conceptual third party risk model for personal and unmanned aerial vehicles. *2015 International Conference on Unmanned Aircraft Systems (ICUAS)*, 1301–1309. https://doi.org/10.1109/ICUAS.2015.7152424

Ale, B., & Piers, M. (2000). The assessment and management of third party risk around a major airport. *Journal of Hazardous Materials*, *71*(1), 1–16. https://doi.org/10.1016/S0304-3894(99)00069-2

Ancel, E., Capristan, F. M., Foster, J. V., & Condotta, R. C. (2017). Real-time risk assessment framework for unmanned aircraft system (UAS) traffic management (UTM). *17th AIAA Aviation Technology, Integration, and Operations Conference*. https://doi.org/10.2514/6.2017-3273

ARC. (2015). Unmanned aircraft systems (UAS) registration task force (RTF) aviation rulemaking committee (ARC).

Austin, J., & Brewster, S. (2024). *The 4 best drones for photos and video of 2024 — reviews by wirecutter* [The best drones for photos and video]. Retrieved July 12, 2024, from https://www.nytimes.com/wirecutter/reviews/best-drones/

Ball, J. A., Knott, M., & Burke, D. D. (2012). CRASH LETHALITY MODEL.

Barr, L. C., Newman, R., Ancel, E., Belcastro, C. M., Foster, J. V., Evans, J., & Klyde, D. H. (2017). Preliminary risk assessment for small unmanned aircraft systems. *17th AIAA Aviation Technology, Integration, and Operations Conference*. https://doi.org/10.2514/6.2017-3272

Bleier, M., Settele, F., Krauss, M., Knoll, A., & Schilling, K. (2015). Risk assessment of flight paths for automatic emergency parachute deployment in UAVs. *IFAC-PapersOnLine*, *48*(9), 180–185. https://doi.org/10.1016/j.ifacol.2015.08.080

Breunig, J., Forman, J., Sayed, S., & Audenaerd, L. (2018). Modeling risk-based approach for small unmanned aircraft systems.

Burke, D. A., Hall, C. E., & Cook, S. P. (2011). System-level airworthiness tool. *Journal of Aircraft*, *48*(3), 777–785. https://doi.org/10.2514/1.C031022

Clothier, R., Walker, R., Fulton, N., & Campbell, D. (2007). A CASUALTY RISK ANALYSIS FOR UNMANNED AERIAL SYSTEM (UAS) OPERATIONS OVER INHABITED AREAS.

Cole, J. K., Young, L. W., & Jordan-Culler, T. (1997, April 1). *Hazards of falling debris to people, aircraft, and watercraft* (SAND-97-0805). Sandia National Lab. (SNL-NM), Albuquerque, NM (United States). https://doi.org/10.2172/468556

Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2022). *Introduction to algorithms*. MIT press.

Cour-Harbo, A. l. (2020). Ground impact probability distribution for small unmanned aircraft in ballistic descent. *2020 International Conference on Unmanned Aircraft Systems (ICUAS)*, 1442–1451. https://doi.org/10.1109/ICUAS48674.2020.9213990

Dalamagkidis, K., Valavanis, K. P., & Piegl, L. A. (2008). Evaluating the risk of unmanned aircraft ground impacts. *2008 16th Mediterranean Conference on Control and Automation*, 709–716. https://doi.org/10.1109/MED.2008.4602249

Dask Development Team. (2016). *Dask: Library for dynamic task scheduling*. http://dask.pydata.org

Dijkstra, E. W. (2022). A note on two problems in connexion with graphs. In *Edsger wybe dijkstra: His life, work, and legacy* (pp. 287–290).

DJI. (2023). *DJI agriculture - drones better growth, better life*. Retrieved July 12, 2024, from https://ag.dji.com/

DJI. (2024). *Consumer drones comparison - DJI* [Consumer drones comparison]. Retrieved July 12, 2024, from https://www.dji.com/id/products/comparison-consumer-drones

Dorigo, M. (2007). Ant colony optimization. *Scholarpedia*, *2*(3), 1461.

FAA. (2000, August 20). *EXPECTED CASUALTY CALCULATIONS FOR COMMERCIAL SPACE LAUNCH AND REENTRY MISSIONS*. U.S Department of Transportation. https://www.faa.gov/about/office_org/headquarters_offices/ast/licenses_permits/media/Ac4311fn.pdf

FAA. (2016). eCFR :: 14 CFR part 107 – small unmanned aircraft systems (FAR part 107). Retrieved July 12, 2024, from https://www.ecfr.gov/current/title-14/chapter-I/subchapter-F/part-107

Global human settlement - mission - european commission. (2023). Retrieved May 8, 2023, from https://ghsl.jrc.ec.europa.eu/about.php

Gonzales, E., & Murray, D. (2017). *FAA's approaches to ground and NAS separation distances for commercial rocket launches* [Aerospace sciences meetings] [Archive Location: world]. https://doi.org/10.2514/6.2010-1540

Harwick, M. J., Hall, J., Tatom, J. W., & Baker, R. G. (2007, February 2). *Approved methods and algorithms for DoD risk-based explosives siting:* Defense Technical Information Center. Fort Belvoir, VA. https://doi.org/10.21236/ADA463571

Holland, J. (2015). *How drones are affecting wildlife in surprising ways*. Retrieved July 12, 2024, from https://www.nationalgeographic.com/animals/article/150825-drones-animals-wildlife-bears-science-technology

Kim, Y., & Bae, J. (2022). Risk-based UAV corridor capacity analysis above a populated area. *Drones*, *6*(9), 221. https://doi.org/10.3390/drones6090221

Klepeis, N. E., Nelson, W. C., Ott, W. R., Robinson, J. P., Tsang, A. M., Switzer, P., Behar, J. V., Hern, S. C., & Engelmann, W. H. (2001). The national human activity pattern survey (NHAPS): A resource for assessing exposure to environmental pollutants. *Journal of Exposure Science & Environmental Epidemiology*, *11*(3), 231–252. https://doi.org/10.1038/sj.jea.7500165

Knuth, D. E. (1997). *The art of computer programming, volume 1 fundamental algorithms*. Addison Wesley Longman Publishing Co., Inc.

la Cour-Harbo, A. (2019). Quantifying risk of ground impact fatalities for small unmanned aircraft. *Journal of Intelligent & Robotic Systems*, *93*(1), 367–384. https://doi.org/10.1007/s10846-018-0853-1

LaValle, S. M. (2006). *Planning algorithms*. Cambridge university press.

Magister, T. (2010). The small unmanned aircraft blunt criterion based injury potential estimation. *Safety Science*, *48*(10), 1313–1320. https://doi.org/10.1016/j.ssci.2010.04.012

Melnyk, R., Schrage, D., Volovoi, V., & Jimenez, H. (2014). A third-party casualty risk model for unmanned aircraft system operations. *Reliability Engineering & System Safety*, *124*, 105–116. https://doi.org/10.1016/j.ress.2013.11.016

Nemire, B. (2015). *DJI launches GPU-based high performance embedded computer for drones — NVIDIA technical blog*. Retrieved July 20, 2024, from https://developer.nvidia.com/blog/dji-launches-gpu-based-high-performance-embedded-computer-for-drones-2/

OpenStreetMap wiki. (2022). Retrieved September 21, 2022, from https://wiki.openstreetmap.org/wiki/Main_Page

Pat-Cornell, M. E. (1996). Uncertainties in risk analysis: Six levels of treatment.

Primatesta, S., Guglieri, G., & Rizzo, A. (2019). A risk-aware path planning strategy for UAVs in urban environments. *Journal of Intelligent & Robotic Systems*, *95*(2), 629–643. https://doi.org/10.1007/s10846-018-0924-3

Primatesta, S., Rizzo, A., & la Cour-Harbo, A. (2020). Ground risk map for unmanned aircraft in urban environments. *Journal of Intelligent & Robotic Systems*, *97*(3), 489–509. https://doi.org/10.1007/s10846-019-01015-z

Primatesta, S., Scanavino, M., Guglieri, G., & Rizzo, A. (2020). A risk-based path planning strategy to compute optimum risk path for unmanned aircraft systems over populated areas. *2020 International Conference on Unmanned Aircraft Systems (ICUAS)*, 641–650. https://doi.org/10.1109/ICUAS48674.2020.9213982

Russell, S. J., & Norvig, P. (2016). *Artificial intelligence: A modern approach*. Pearson.

Russell, S. J., Norvig, P., & Davis, E. (2010). *Artificial intelligence: A modern approach* (3rd ed). Prentice Hall.

Shelley, A. (2016). A model of human harm from a falling unmanned aircraft: Implications for UAS regulation. *International Journal of Aviation, Aeronautics, and Aerospace*. https://doi.org/10.15394/ijaaa.2016.1120

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., et al. (2016). Mastering the game of go with deep neural networks and tree search [Publisher: Nature Publishing Group]. *nature*, *529*(7587), 484–489.

Small, U. A. S. (2009). Aviation rulemaking committee (ARC). *Comprehensive Set of Recommendations for sUAS Regulatory Development*, *1*.

The Apache Software Foundation. (2024). *SparkR: R front end for 'apache spark'*. https://www.apache.org%20https://spark.apache.org

Washington, A., Clothier, R. A., & Silva, J. (2017). A review of unmanned aircraft system ground risk models. *Progress in Aerospace Sciences*, *95*, 24–44. https://doi.org/10.1016/j.paerosci.2017.10.001

Weibel, R. E., & Hansman, R. J. (2012). Safety considerations for operation of different classes of UAVs in the NAS.