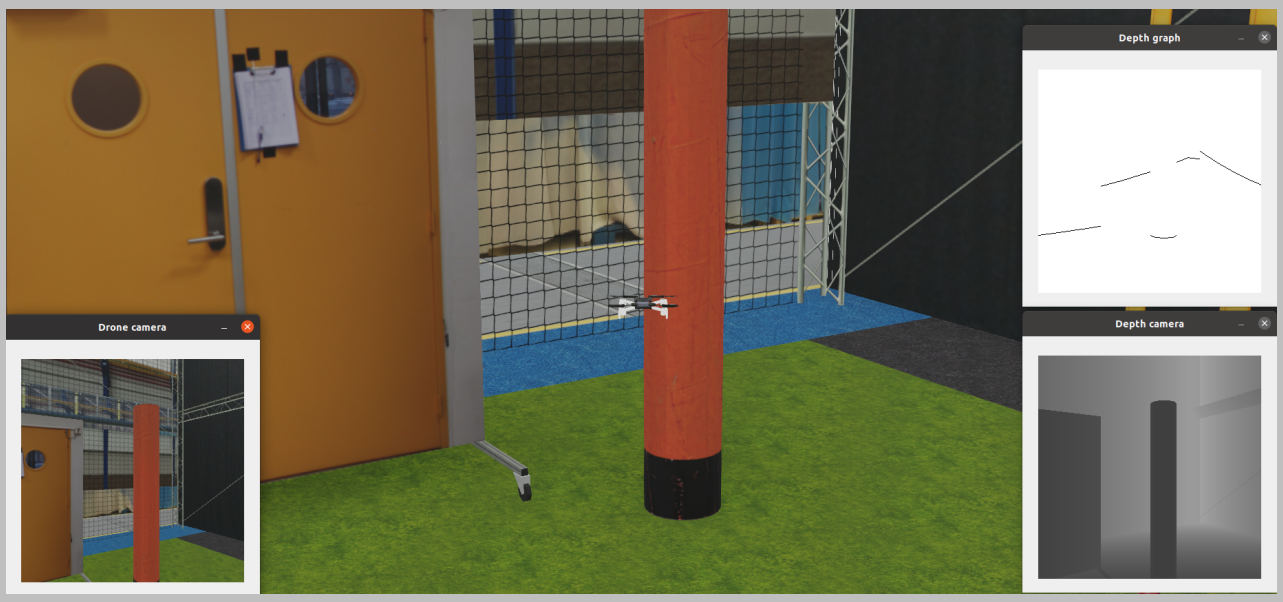
Real-time obstacle avoidance for micro aerial vehicles in 2D and 3D narrow spaces.



Wim N. Pilkes

Layout: typeset by the author using LaTeX.
Cover illustration: Unknown artist

Real-time obstacle avoidance for micro aerial vehicles in 2D and 3D narrow spaces.

Wim N. Pilkes
11044632

Bachelor thesis
Credits: 18 EC

Bachelor *Kunstmatige Intelligentie*

University of Amsterdam
Faculty of Science
Science Park 904
1098 XH Amsterdam

Supervisor
Dr. A. Visser

Informatics Institute
Faculty of Science
University of Amsterdam
Science Park 904
1098 XH Amsterdam

Semester 2, 2022

Abstract

Modern Unmanned aerial vehicles (UAVs) have an increasingly wider range of capabilities. These drones are able to perform tasks that humans cannot do. For instance, in the field of exploration of dangerous and hard to reach areas. As a result of more advanced and compact hardware in addition to continuous advancements in automation, this field of research is expanding rapidly. Even though completely autonomously flying drones are already utilized for an assortment of applications, widespread use of true autonomous drone navigation has not yet been adopted. True autonomy is even harder to achieve when the functioning of a UAV is limited. This is a focus of research in micro aerial vehicles (MAVs). In this field, drones are restricted in processing power and the amount of information that can be observed about its environment. Therefore, efficient path planning algorithms are necessary. This thesis aims to provide upon real-time obstacle avoidance in indoor cluttered environments. This is done by comparing four different lightweight local path planning algorithms. These are both 2D and 3D implementations of the Virtual Force Field algorithm and the Vector Field Histogram algorithm. These methods are tested and compared with a dataset of increasingly more difficult 2D and 3D simulations.

Contents

Chapter 1

Introduction

Autonomous robot navigation is a popular point of discussion. Recent advancements in this field make it an intriguing area of research. This is shown through a variety of novel applications in unmanned aerial vehicle (UAV) technology.

Fer example, drones are used to inspect locations that are inaccessible for humans. This has been applied in the Chernobyl nuclear power plant, where there have been increasing concerns of nuclear waste leaking from one of the reactors since its meltdown. Leaking nuclear waste would have devastating consequences for the containment and clean-up efforts in and around the power plant. Because of the lethal levels of nuclear radiation, humans cannot enter the reactor and asses this damage. As a result, the first successful flight of a quadcopter was done in the reactor in 2020[1]. This drone was able to record footage within the reactor core, and it was determined that there was no leaking nuclear waste.

Another example of the usefulness of quadcopter drones is the inspection of subway tunnel systems. This is already routinely being done in Tokyo[2] and experiments for a similar system are being done in Paris[3]. It saves in manpower and resources, such as the setup of scaffolding. Even more so, after a collapse of a parking garage above a subway line in Boston[4], a drone was utilized to determine the risk of collapse before inspectors were sent in. This eliminated the risk of injury for these inspectors.

[1] https://www.commercialdroneprofessional.com/inside-chernobyl-elios-2-drone-sent-into-reactor-on-inspection-mission/

[2] https://dronedj.com/2020/02/27/drones-inspect-tokyo-subway-tunnels/

[3] https://dronedj.com/2022/06/07/paris-metro-looks-to-modernize-with-drone-inspections/

[4] https://gcn.com/public-safety/2022/04/drone-inspects-subway-tunnels-after-nearby-building-collapse/364049/

Flying drones are also used for science. For example, the European Space Agency (ESA) is using a drone to explore cave systems that humans cannot reach. These caves are either too narrow or deemed to dangerous for humans. The ESA aims to use this technology to do autonomous exploration and research in the future on foreign planets[5].

These examples demonstrate the importance of quadcopter technology. It does not only show how business processes and science can be improved, but also how this technology helps to reduce risk and possibly save lives. However, these drones are more often than not still being operated by a human. Completely autonomous systems are being developed and tested, but are currently rarely used in full scale commercial operations. Furthermore, these examples demonstrate that there is a need for completely autonomously navigating drones to enter locations that humans cannot reach.

Given this wide range of application of drone technology, the field of robotics has a large assortment of different navigation algorithms for numerous purposes. This is due to a number of factors, for example, different types of drone design, different working environments and different drone observation methods (LaValle, 2006, Chapter 1). Other constrains, such as limits in processing power also influence how path navigation is implemented. Nevertheless, robot navigation can be generalized as follows: a robot needs to move from its current position to a goal position as efficient as possible, without hitting any obstacles in its path. To do so, the robot makes observations about its environment and runs this through an algorithm. This results into one or more paths, of which the path that is considered the best is chosen and executed.

With the steady reduction in chip size and the advancements in robotics and navigation algorithms in the last few years, it is possible to create increasingly smaller and more capable self-processing robots. *The International Micro Air Vehicle Conference and Competition* (IMAV)[6] is a yearly event that aims to advance micro air vehicles (MAVs). This conference encompasses both new drone designs and improvements in drone control and path planning. The scientific conference is combined with a set of competitions to test new technologies. One of these competitions is the *nanocopter AI challenge*[7]. This challenge achieves to find solutions for more efficient autonomously flying drones and better path planning and obstacle avoidance in small indoor environments. In the challenge, a nano quadcopter

[5]https://www.esa.int/Science_Exploration/Human_and_Robotic_Exploration/CAVES_and_Pangaea/Explori
[6]https://2022.imavs.org/
[7]https://2022.imavs.org/index.php/competition/nanocopter-ai-challenge/

drone has to navigate an obstacle zone and fly trough a goal as fast as possible, completely autonomously. To do so, the competition sets out a framework for the competitors. This consists of a small indoor area, cluttered with a number of differently shaped obstacles and a square goal. Furthermore, a state of the art open source MAV by Bitcraze[8] is selected, which is able to do real-time image processing, path planning and decision making. A programmable simulation of this framework is also provided. This framework is not only suitable for research in UAVs in indoor enviroments, it enables researchers to compare their implementations[9].

Another focus of this challenge is to determine capabilities of autonomous navigation under limited circumstances. First of all, the drone is limited to on-board processing. Even though the drone has a state of the art processor designed specifically to do complex machine learning, a path navigation algorithm is required to be lightweight enough to compute in real-time. Furthermore, the drone will be limited in what observations it can do. The only inputs that the drone will be given is a monocular monochrome front-facing camera attached to the drone and the data from its internal gyroscope. No other inputs, such as a GPS system or external cameras are allowed. Therefore a perfect mapping of the drones surroundings is not possible. This limits the range of path navigation implementations to lightweight local path planning algorithms (Zhang et al., 2018).

Considering these limitations, two lightweight local path planning algorithms are implemented in the IMAV simulation. These are called the *virtual force field* (Borenstein and Koren, 1989) and the *vector field histogram* (Borenstein and Koren, 1990). Given the real world environment of the IMAV framework, novel three-dimensional variations of the VFF and the VFH algorithms will be developed and tested. This thesis aims to improve upon low computing power path planning capabilities for nanocopter drones in a real world environment. It does this by answering the following research question: "How well do 2D and 3D implementations of the VFF and VFH algorithms perform compare to each other in a narrow cluttered environment?"

The structure of the paper is organized as follows. At first, current capabilities will be determined and relevant work will be summarized. The two different algorithms will be implemented in a simulation. These will then be transformed into two novel algorithms that operate in three dimensions. Furthermore, a dataset

[8]https://www.bitcraze.io/

[9]The UvA drone team will participate in this competition in September 2022 in Delft, the Netherlands. This thesis is part of the preliminary research.

of increasingly difficult simulations will be set out to conduct experiments. This will result in a quantification of the efficiency of the different algorithms, which will be compared to draw conclusions on the performance and limitations of the algorithms.

Chapter 2

Background

In this chapter a theoretical overview will be given of relevant literature. This will serve as the foundation for the research that is presented. First, general theory about path planning is summarized. Secondly, fundamental theory about path planning algorithms is explained. Furthermore, research about relevant algorithms will be described. Finally, two new implementations of algorithms will be theorized, which are able to operate in three dimensions.

2.1 Static and Dynamic Enviroments

Buniyamin et al. (2011) make a distinction between static environments and dynamic environments. An environment is considered static when obstacles do not move. To the contrary, an environment is considered dynamic when the environment is not fully explored or when obstacles move in an unpredictable manner. This is an important factor to determine when a path planning algorithm is chosen, considering that a robot should be able to behave efficiently to its environment.

2.2 Global and Local Path Planning

Robot path planning algorithms are categorized in two groups, global path planning and local path planning (Zhang et al., 2018). The difference between these two categories is defined by whether the algorithm has complete information about its environment. Therefore, the amount of observation information given to a robot will determine what category of path planning is suitable.

2.2.1 Global path planning

Global path planning requires a complete mapping of an environment before execution of the algorithm. This information is loaded into the algorithm and a full path from start to goal is calculated (Giesbrecht, 2004). Then, the robot is instructed to follow this path. If at least one path between a start and a goal exists, this category of path planning will guarantee to find a successful path.

However, global path planning algorithms cannot function in unknown environments. Furthermore, these algorithms have difficulty adapting to dynamic environments. Only when trajectories of obstacles are known beforehand, then global path planning algorithms can be designed to adapt to this. When this is not known, these algorithms require a completely new path evaluation for each change in an environment (Raja and Pugazhenthi, 2012).

2.2.2 Local path planning

Local path planning, or real-time obstacle avoidance, does not require complete information about the environment. A local path planning algorithm uses sensory data to make observations about a segment of the environment, localized around the robot. This is executed in real-time. As a result, local path planning is able to navigate in unknown and changing environments. Furthermore, it is able to avoid dynamic obstacles (Khaksar et al., 2015). In general, when no obstacle is detected, a local path planning attempts to move towards a predetermined goal in a straight line. It will deviate from this line to avoid a detected obstacle (Buniyamin et al., 2011).

However, this category of path planning has certain limitations. First of all, the incompleteness of information can make it difficult to identify a goal. If the robot cannot observe a goal, the robot cannot determine where to move towards. Thus failing the planning algorithm. Second of all, the algorithm is prone to local minima . A local minimum is a situation in which a robot is trapped in a position (Borenstein and Koren, 1989). This happens when a robot encounters one or more obstacles that are configured in a manner that no viable path can be found. In such a situation, any movement forwards or sideways will move the robot too close to an obstacle, resulting in the robot moving back to the local minimum. Any movement backwards will result in the robot moving further away from the goal and free itself from the trap. It will then continue moving forwards towards the goal and re-enter the trap. Raja and Pugazhenthi (2012) reviewed a number of different approaches to detect and exit local minima.

2.3 Potential Field Methods

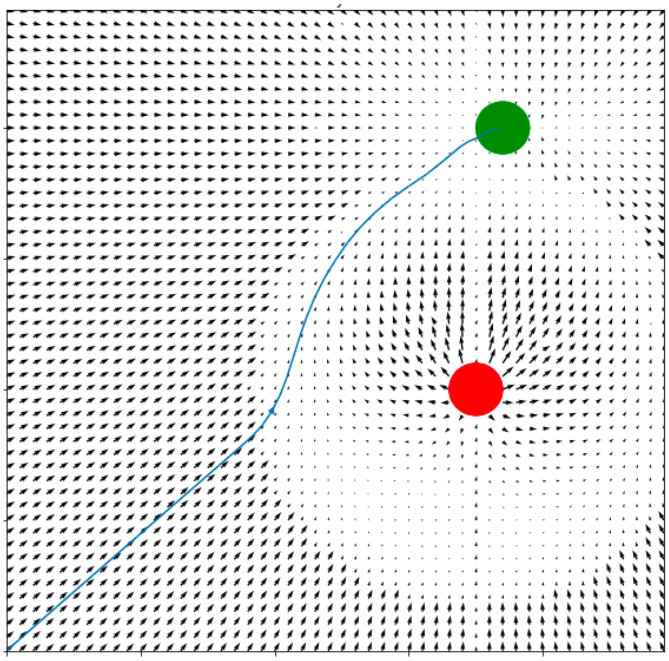


Figure 2.1: A fully calculated potential field. Each vector represents the potential at its corresponding position. These potentials are calculated by the repulsing force generated by the obstacle (red) and the attracting force generated by the goal (green). The robot starts at the bottom left. For each position, the corresponding potential is applied to the robot. Resulting in a path (blue) towards the goal that avoids the obstacle.

Potential field methods (PFMs) execute path navigation by considering a robot as a particle that can be influenced by a potential field (Sabudin et al., 2016). This field is determined by repulsing potentials and an attracting potential. Within this field, obstacles are emitting repulsing potentials and the goal is emitting an attracting potential. This potential will move the robot. Moreover, the motion and direction of the drone is calculated by the net force of this potential on any given moment, see Figure 2.1. This results in a path in which the drone is attracted towards the goal, whilst being repulsed by obstacles. Khatib (1985) calculated the artificial potential for each position a robot can have within an environment. This created a global path planning algorithm for a static environment. This makes it

unsuitable for real-time autonomous obstacle avoidance.

However, this method can be transformed in a local path planning algorithm by only considering obstacles in the direct proximity of a robot and by continuously recalculating the potential field with only these obstacles and the goal. Therefore, only calculating the potential at the exact location of the robot. This results in algorithms such as, the virtual force field and the vector field histogram.

2.3.1 Virtual Force Fields

First used by Borenstein and Koren (1989), a virtual force field (VFF) is a local path planning algorithm. This method builds upon the concept of attracting and repulsing forces by reducing the potential field to a small area, centered around the robot. Because the robot moves, and this field moves with it, the field is constantly updated. This allows the robot to work with incomplete knowledge of the environment, making it a local path planning algorithm. Furthermore, since the field is constantly recalculated, the robot can operate in a dynamic environment.

This method is further improved by transforming the potential field into a certainty grid (Moravec, 1987). Here, each cell within the grid has a certainty value. As described by Moravec and Elfes (1985) and by Elfes (1986), for each cell, the certainty value is defined as the probability that there is an object in the cell. These probabilities are estimated by the observations of the robot, such as sonar data. Each cell within the grid is converted into a force vector. This vector is proportional to the certainty value. Therefore, cells with a high certainty value result in a stronger force. Furthermore, the vector is inversely proportional to the distance, cells closer to the robot result in stronger force. This makes the obstacle avoidance more sensitive to objects close to the robot . The strength and direction towards the robot of each force are added together. This results in a net force that acts upon the drone. The force vector F for each cell $(i, , j)$ is defined as follows (Borenstein and Koren, 1988):

$$F(i,j) = \frac{F_{cr} C(i,j)}{d^2(i,j)} \left[\frac{x_i - x_o}{d(i,j)} \hat{x} + \frac{y_j - y_o}{d(i,j)} \hat{y} \right] \tag{2.1}$$

Here, F_{cr} is the force constant, a hyper-parameter to adjust the sensitivity of the repelling force. $C(i,j)$ is the certainty value of the cell. $d(i.j)$ is the euclidean distance between the robot and the cell. x_i and y_j are the coordinates of the cell, and x_0 and y_0 are the coordinates of the robot. The sum of these forces result in a single force that acts upon the robot:

$$F_r = \sum_{i,j} F(i.j) \tag{2.2}$$

This repulsive force will ensure that the robot will avoid obstacles, but will not steer the robot towards a desired goal. Thus, an attracting force is necessary. This is defined as a single cell of which the coordinates are known. The attracting force F_t can be calculated similarly.

$$F_t = F_{ct} \left[\frac{x_t - x_o}{d(t)} \hat{x} + \frac{y_t - y_o}{d(t)} \hat{y} \right] \tag{2.3}$$

Here, F_{ct} is the force constant, d_t is the distance between the goal and the robot, and x_t and y_t are the coordinates of the robot. The sum of the repulsing force and the attracting force result in a net force vector R that acts upon the robot:

$$R = F_t + F_r \tag{2.4}$$

For each iteration of the algorithm this net force will be exerted upon the robot.

2.3.2 Vector Field Histograms

The VFF has a high level of data reduction. All the certainty values in the grid are reduced in one step to a single vector and a lot of features about the environment are lost (Koren et al., 1991). As a result, a robot using a VFF has issues with detecting narrow passages or navigating through cluttered environments (Oroko and Nyakoe, 2022). Borenstein and Koren (1990) propose to change the single-stage data reduction into a two-stage data reduction to solve these problems, creating the vector field histogram (VFH).

The first stage of the data reduction is similar to the first steps of the VFF. A grid is defined around the robot and, using the robots observations, certainty values are determined for each cell in the grid. However, the robot is set to do rapid range readings of a 1 cell wide range, to reduce the amount of observations and computations that are necessary. Each individual sample is used to increment a certainty value corresponding to the position of the range reading. This creates a one dimensional polar histogram, consisting of certainty based obstacle vectors. Each vector $c^*(i,j)$ represents the set of range readings in that direction $\beta(i,j)$. Directions in which obstacles are being observed often, have an continuously increasing magnitude $m(i,j)$:

$$m(i,j) = (c^*(i,j))^2 * [a - b * d(i,j)] \tag{2.5}$$

Here, a and b are positive constants. and $d(i,j)$ is the distance from the cell to the robot.

Furthermore, this histogram grid is not used to create a single repulsing force vector. The grid is divided in k evenly spaced sectors around the drone. The magnitudes of the cells within that sector is summed, creating a polar obstacle density value for each sector h_k:

$$h_k = \sum_{i,j} m(i,j) \tag{2.6}$$

This results in a force vector for each sector. To reduce noise and improve efficiency, this histogram grid is smoothed and only values above a predefined threshold are considered.

The second stage of data reduction determines which sector is the optimal path for the robot to navigate towards. To do so, it first creates a set of candidate valleys. Which are sections within the histogram grid that are below the threshold. These valleys can be one or more sections wide, were valleys with a low number of sectors is considered as "narrow" and valleys with a high number of sectors is considers "wide". The VFH algorithms then selects the valley that matches its direction of travel the closest to the direction of the target the robot is intended to move towards. Then, the angle of most optimal sector of this valley is the new direction of the robot. This is the middle sector for both narrow and wide valleys. However, when a wide valley is chosen that encounters an obstacle, the robot is instructed to travel along that obstacle from a predetermined distance. This reduces steering fluctuations (Borenstein et al., 1991).

2.3.3 Path Finding in 3D

The VFF and VFH as described above operate in a two dimensional environment. However, some research has been done in potential field methods in three dimensional environments. For example, Saravanakumar and Asokan (2013) apply a three dimensional PFM to an autonomous underwater vehicle. This is done by adding a height vector $\hat{z}$ to the potential field.

This method be used to create a 3D VFF. A force vector for a cell, described in equation (2.1), is defined as follows:

$$F(i,j,k) = \frac{F_{cr}C(i,j,k)}{d^2(i,j,k)} \left[\frac{x_i - x_o}{d(i,j,k)}\hat{x} + \frac{y_j - y_o}{d(i,j,k)}\hat{y} + \frac{z_j - z_o}{d(i,j,k)}\hat{z} \right] \tag{2.7}$$

The total repulsive force, described in equation (2.2), is defined as follows:

$$F_r = \sum_{i,j,k} F(i.j,k) \tag{2.8}$$

When this repulsive force changes the height of the drone, a counteracting force is needed to bring the drone back to the height of the goal. Therefore, the attracting force, as described in (2.3) should also be transformed:

$$F_t = F_{ct}\left[\frac{x_t - x_o}{d(t)}\hat{x} + \frac{y_t - y_o}{d(t)}\hat{y} + \frac{z_t - z_o}{d(t)}\hat{z}\right] \tag{2.9}$$

Given that both the repulsing and the attracting forces are three dimensional, the resulting force, defined by equation (2.4) would also be in 3D.

Furthermore, this conversion into a 3D algorithm can also be done for the VFH. Adding height to this algorithm would make polar histogram a 2D object. The magnitude, defined in equation (2.5) and the polar obstacle density, defined in (2.6) will require a height dimension k:

$$m(i,j,k) = (c^*(i,j,k))^2 * [a - b * d(i,j,k)] \tag{2.10}$$

$$h_k = \sum_{i,j,k} m(i,j,k) \tag{2.11}$$

This chapter has provided an overview of path planning algorithms, with a focus on local path planning. Two different 2D local path planning algorithms were described and two novel 3D implementations of these algorithms were theorized.

Chapter 3

Method

To be able to test the abilities of the algorithms described in the background, a relevant environment is necessary. In this chapter, a real-time obstacle avoidance competition is chosen as environment. This chapter describes this platform and how the four local path planning algorithms are implement and evaluated.

3.1 Platform

The 2022 edition of the International Micro Air Vehicle Conference and Competition outlines a set of requirements for the IMAV Nanocopter AI challenge[1]. It is the goal of this challenge to be able to navigate a small obstacle zone as efficient as possible without hitting the obstacles. Furthermore, this has to be accomplished with real-time processing on a MAV that is only able to make observations through an onboard camera. These factors make it a suitable testing environment for the research presented in this thesis.

This challenge requires all participants to use identical hardware. This consists of the AI deck 1.1 attached to the Crazyflie 2.1 nano quadcopter, both are produced by Bitcraze. The competition will be held in an arena named the "Cyberzoo". An accurate simulation of the nanocopter and the cyberzoo is provided in the webots 3D robot simulator.

[1] https://2022.imavs.org/index.php/competition/nanocopter-ai-challenge/

Figure 3.1: The Crazyflie 2.1 nano quadcopter.

Figure 3.2: The AI-deck 1.1.

3.1.1 The Crazyflie 2.1 Nano Quadcopter

The Crazyflie 2.1 is an open source flying development platform designed and produced by Bitcraze, see Figure 3.1. This platform allows for efficient development and testing of autonomous drone software. The platform consist of an control board, a small LiPo battery, 4 DC coreless motors with motor mounts and propellers. This platform is designed to support expansion decks for further functionality, such as the AI-deck 1.1.

3.1.2 AI-deck 1.1

Considering that the drone is limited in size and required to run image processing and path navigation algorithms without off-board processing, a lightweight and powerful processor is required. Palossi et al. (2019) used the *GreenWaves GAP8 IoT application processor*[2] for this purpose. The boards 8 cores and 1 microprocessor produce a processing power of $22.65GOPS$ at a power consumption of $4.24mW/GOP$ [3]. Furthermore, the processors architecture is specifically designed to do efficient machine learning. For example, it contains a Hardware Convolution Engine (HWCE), a convolutional neural network accelerator. This processor was mounted on a Crazyflie 2.1. To do so, a shield was created that connects the processor to the Crazyflie header pins, allowing it to be attached above or under

[2]https://greenwaves-technologies.com/gap8$_m cu_a i$/

[3]https://greenwaves-technologies.com/wp-content/uploads/2021/04/Product-Brief-GAP8-V1_9.pdf

the Crazyflie. In addition, a camera was added to the shield.

This prototype resulted in the *AI-deck 1.1* [4], a commercially available platform by Bitcraze, see Figure 3.2. The deck has an 320x320 pixel monochromatic camera, directly attached to the processor.

3.1.3 The Cyberzoo

To be able to test the nanocopter in a obstacle filled space, an 8 meter by 8 meter arena is provided by the competition. This obstacle zone contains a yellow square goal and a set of cylindrical poles and grey rectangular panels. The Crazyflie is placed within this zone, in which it will fly autonomously towards the goal whilst avoiding the obstacles.

3.2 The Simulated Environment

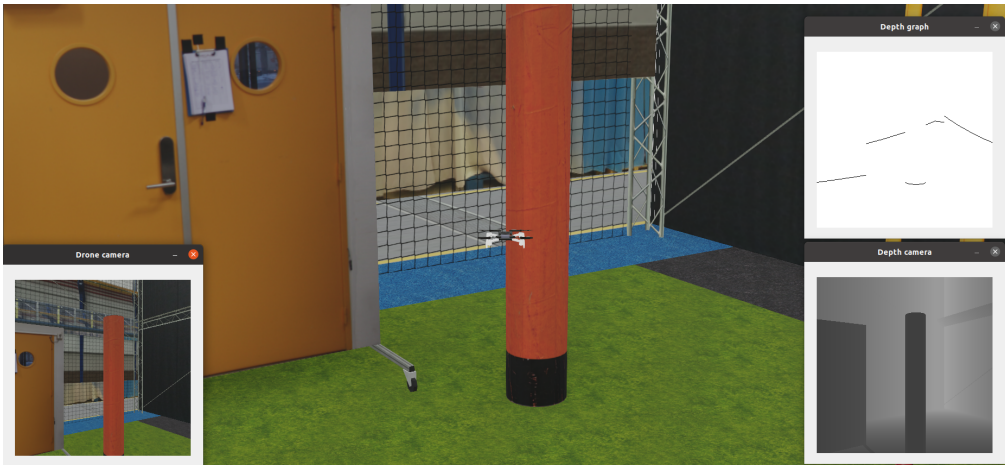


Figure 3.3: An example of the Webots Cyberzoo simulation. The Crazyflie nanocopter is showed in the center of the simulated environment. On the bottom-left the real-time feed from the camera in the nanocopter is displayed. On the bottom-right the real-time feed from the depth sensor in the nanocopter is displayed. A horizontal slice from the middle of this feed is used to create a representation of the depth readings of the different objects (upper-right).

A complete simulation of the Cyberzoo in the Webots simulation software is provided by the competition, see Figure 3.3. This simulates the arena in a physics

[4] https://store.bitcraze.io/products/ai-deck-1-1

engine. The objects and goal are also simulated. Furthermore, the Crazyflie quadcopter is also simulated, using a built-in PID controller for stabilization. Furthermore, the quadcopter simulates a front facing 320x320 pixel camera, identical to the real-world Crazyflie.

The simulation is altered to be able to accommodate the path planning algorithms. First of all, the camera is converted into a depth sensor. Instead of directly observing its environment, the quadcopter observes a distance value for each pixel. Second of all, the distance and angle between the goal and the drone is calculated. The algorithms in this thesis require these observations to do path planning. Implementing a depth estimation and a goal detection algorithm is beyond the scope of this thesis.

3.2.1 Dataset Generation

To be able to create a significant sample size, more than one simulated environment is necessary. Therefore, the simulation is modified into a set of randomly generated simulations. These simulations have increasingly more obstacles, resulting in different difficulty levels. Furthermore, a dataset is generated to simulate a 2D environment and a dataset is generated to simulated a 3D environment. This allows for comparison between the VFF and VFH and for comparison between the 2D and the 3D implementations of these algorithms.

For each simulation, the Crazyflie is positioned randomly on one end of the arena, and the goal is positioned randomly on the other side of the simulation. This avoids simulations in which the goal and drone are placed in close proximity to each other. Therefore, the drone is forced to avoid obstacles to reach the goal.

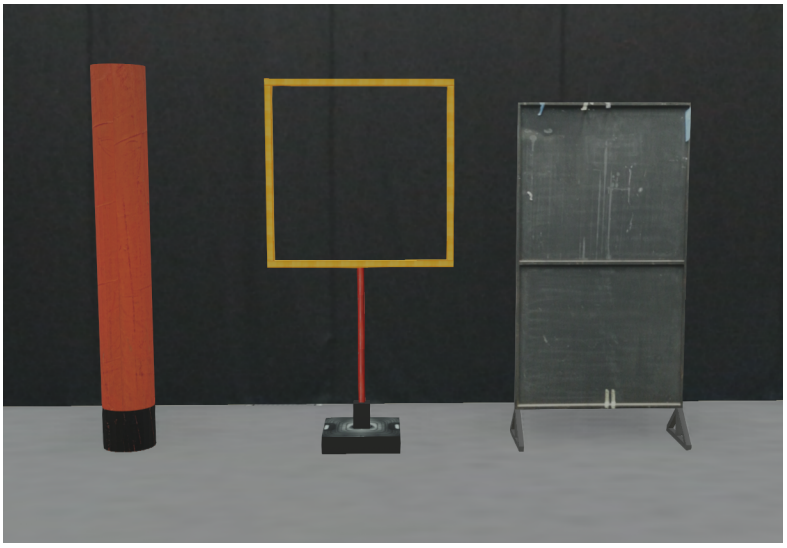


Figure 3.4: Objects used in the Cyberzoo simulation: a cylindrical pole (left), a square gate (middle) and a rectangular grey panel (right).

Next, the obstacles are generated at random positions between the quadcopter and the goal. As seen in Figure 3.4, two types of obstacles are simulated, a cylindrical pole and a rectangular panel. A total of 50 simulations are generated with five difficulty levels. The first set of generated simulations will have two poles and two grey panels. After every tenth simulation, two more poles and two more panels will be generated, resulting in the five difficulty levels. Another dataset of 50 simulations will be made with these same requirements, but half the panels are floating 1.5 meters above the ground. This is to test the capabilities of the 3D algorithms. Resulting in the following overview:

2D Cyberzoo simulations, 10 for each difficulty

Difficulty	Number of Poles	Number of Panels (Normal)	Number of Panels (Floating)
1	2	1	1
1	2	2	0
2	4	4	0
3	6	6	0
4	8	8	0
5	10	10	0

3D Cyberzoo simulations, 10 for each difficulty

Difficulty	Number of Poles	Number of Panels (Normal)	Number of Panels (Floating)
1	2	1	1
2	4	2	2
3	6	3	3
4	8	4	4
5	10	5	5

3.2.2 Evaluation Methods

This dataset is generated to test multiple path planning algorithms. However, a suitable metric is required to be able to evaluate these algorithms. Therefore the simulation logs information about each simulation run for each algorithm.

First of all, whether or not the drone reaches the goal is determined. This is a binary value, 0 means that the path planning failed and 1 means that the goal is reached. If the drone is trapped in a local minimum, then the simulation is considered as failed. This is used to determine the reliability of each algorithm. Furthermore, the simulation is considered as a fail after 120 seconds of runtime. Second of all, the total distance travelled by the robot is tracked for each simulation. And finally, the total runtime for each simulation is tracked. The distance and time are used to determine the efficiency of the path planning of each algorithm. Given that failed simulations can generate outliers in the distance and time values, only successful simulations are used.

The four local path planing algorithms are implemented in python as stated in the chapter background. Parameter optimization was done empirically.

Chapter 4

Results

In this chapter, the results of this thesis are presented. These results are the outcome of 100 simulations for each of the four different real-time obstacle avoidance algorithms. 50 of those simulations were done in 2D and 50 simulations were done in 3D. Both these datasets were further divided in subsets of 10 simulations with an increasing amount of obstacles. Each simulation resulted in three datapoints, whether the simulation was successful, how much distance the drone covered, and how much time it took. For the distance and time values only successful simulations were considered. These three types of results will be evaluated separately.

4.1 Success Rate Evaluation

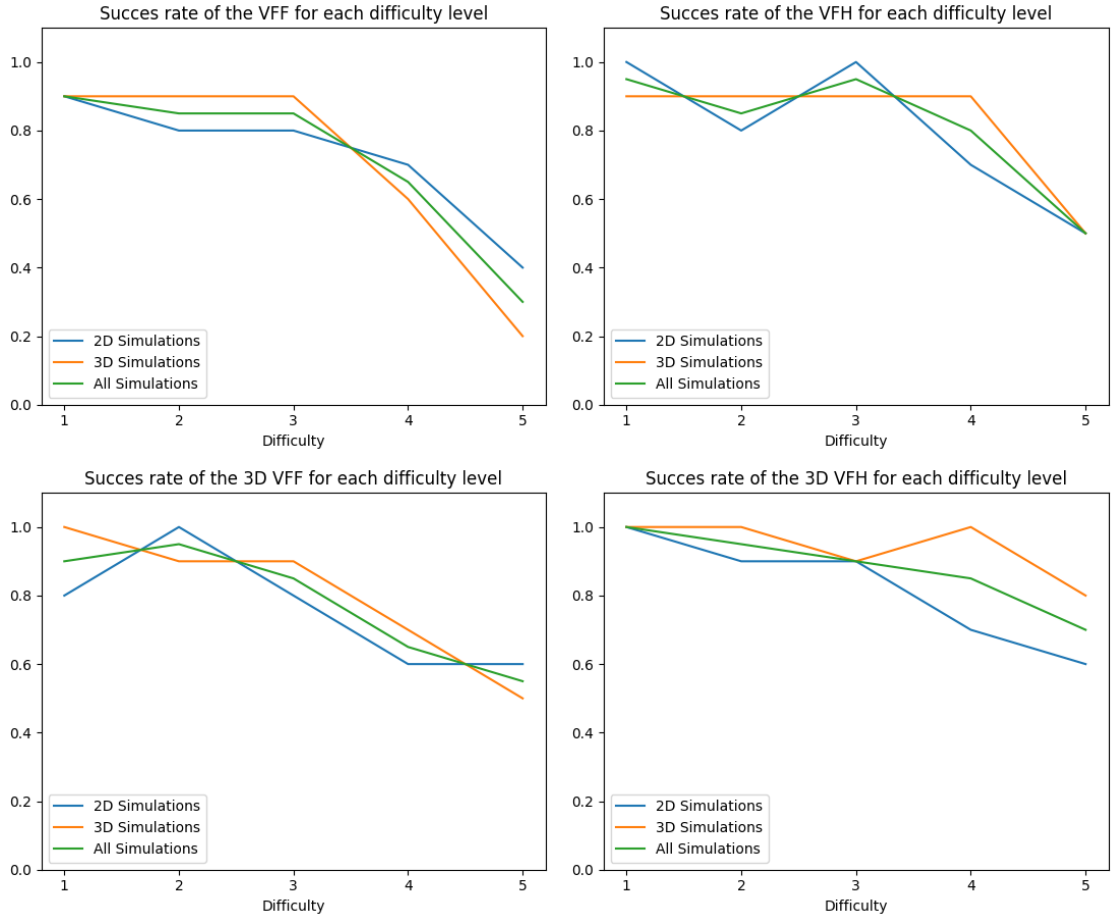


Figure 4.1: The succes rates of the different algorithms. With the 2D VFF algorithm on the top left, the 2D VFH algorithm on the top right, the 3D VFF algorithm on the bottom left and the 3D VFH algorithm on the bottom right.

As seen in Figure 4.1, a relation can be observed between the success rate and the difficulty level for all four the algorithms. An increase in the amount of obstacles result in a decrease of success rate. However, a noticeable drop-off in performance is visible in the 2D implementations at difficulty level 4 and 5. Furthermore, the VFH appear to perform better, maintain a success rate of more than 0.5 for each set of simulations.

4.2 Distance Evaluation

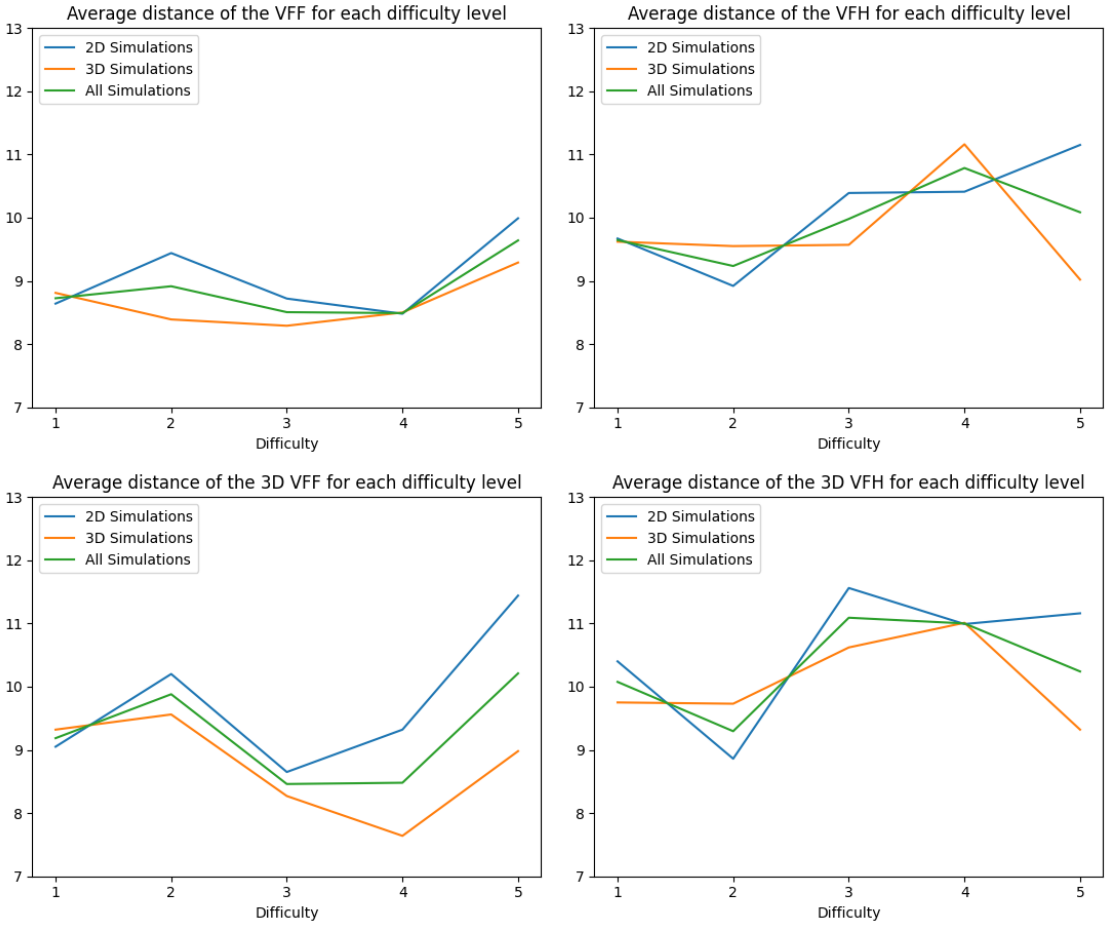


Figure 4.2: The average distance covered by the drone for each successful simulation of the different algorithms. With the 2D VFF algorithm on the top left, the 2D VFH algorithm on the top right, the 3D VFF algorithm on the bottom left and the 3D VFH algorithm on the bottom right.

Figure 4.2 shows that the distance travelled by the different path navigation algorithms are similar. Therefore, it is difficult to evaluate and compare these algorithms.

4.3 Time Evaluation

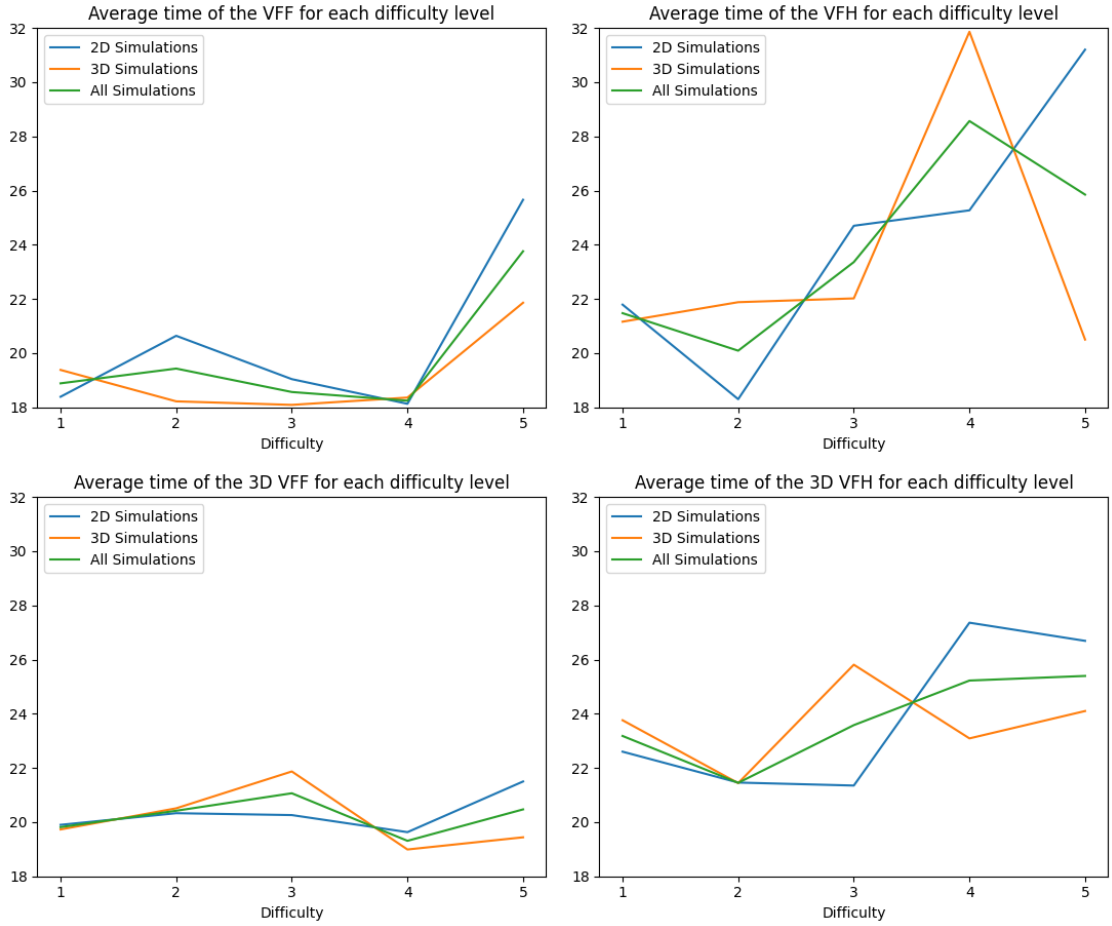


Figure 4.3: The average runtime in seconds for each successful simulation of the different algorithms. With the 2D VFF algorithm on the top left, the 2D VFH algorithm on the top right, the 3D VFF algorithm on the bottom left and the 3D VFH algorithm on the bottom right.

As shown in Figure 4.3, a relation can be observed between the runtime and the difficulty level for all four the algorithms. In which simulations with more obstacles have a higher runtime. Particularly, the 2D VFF takes a long time to complete simulations of difficulty level 5. However, both the VFF algorithms appear to have a more consistent relation between time and difficulty. Whilst the VFH algorithms seem to behave more irregular.

4.4 Further Observations

During the running of the algorithms some notable observations were made. First of all, the VFH algorithms often went through narrow passages, whilst the VFF algorithms would not. Second of all, the 3D algorithms were able to fly over and under the obstacles in the 3D simulations, creating a easier path to the goal.

Although, the results show differences between the performances of the path planning algorithms, there is a significant amount of overlap of results. Therefore, it is not possible to conclusively quantify how the algorithms perform in contrast to each other.

Chapter 5

Discussion

In this chapter, the research done in this thesis will be summarized. The results will be evaluated to answer the research question. Furthermore, future work will be discussed.

5.1 Evaluation

This thesis aimed to answer the following research question: "How well do 2D and 3D implementations of the VFF and VFH algorithms perform compare to each other in a narrow cluttered environment?"

Even though, this thesis was not able to quantify this answer, some notable observations were done. First of all, the VHF has a more consistent success rate than the VFF. Second of all, the implementation of 3D path planning algorithms will allow a drone to more successfully navigate environments with a high amount of obstacles. Furthermore, as described by Koren et al. (1991) the VFH is able to successfully navigate through narrow spaces, which the VFF cannot.

5.2 Future work

A number of recommendations can be made to improve upon this thesis. First of all, the drone simulation can be improved. A real-time local goal detection algorithm could be implemented. If, for example, the goal would be behind an obstacle, a method would have to be created to first search for the obstacle. This would result in an algorithm that can operate in even more circumstances. Additionally, the grayscale camera can be used as input for the local path planning algorithms. To do so, a depth estimation algorithm will have to be implemented,

instead of using the perfect depth camera in the simulation. This could be used to examine how the algorithms would perform with imperfect input. Furthermore, local goal detection and depth estimation would make the whole system completely autonomous and would allow the algorithm in real-life situations. Another problem that could be improved upon, is that local path planning algorithms are prone to local minima. A variety of different local minimum detection algorithms exist (Raja and Pugazhenthi, 2012). Implementing such an algorithm could further improve the capabilities of the path planning.

Secondly, a larger dataset can be generated to find more conclusive results. The results presented in this thesis had a significant amount of overlap. More simulations under a wider variety of different environments could result in a decisive evaluation.

Furthermore, it would be interesting how global path planning would compare with local path planning within this environment. Because this experiment is done in a simulation, this comparison is possible. This would not only allow conclusions to be made between different local path planning algorithms under different conditions, but it could also draw conclusions about the efficiency of local path planning in general.

Finally, the algorithms can be tested outside of the simulation, for instance in the real-world environment of the IMAVs competition in Delft. This would determine whether the simulation is applicable in real-life. Furthermore, other narrow environments, such as metro tunnels or cave systems could be tested on.

5.3 Conclusion

Local path planning is designed to operate in a variety of real-world implementations. It can be used on a variety of robots utilizing a range of observation methods in dynamic environments. Furthermore, creating a system with severe constrains demonstrate the strength and versatility of local path planning. The limitations within this thesis were inspired by the nanocopter AI challenge by IMAV. This challenge was described as follows, a MAV was used with an on-board processor and a small greyscale camera. This drone had to navigate an small indoor obstacle space, filled with obstacles. The aim was to fly the drone through the obstacle zone and reach a goal.

This thesis compared four local path planning algorithms. The aim was to determine how these compared under different conditions. To do so, a simulation of the real-life framework by IMAV was used. Two datasets of increasingly more cluttered simulation worlds were created. One dataset that described a 2D environment and one in 3D. The algorithms were run on these datasets and the performance was evaluated.

References

Borenstein, J. and Koren, Y. (1988), High-speed obstacle avoidance for mobile robots, *in* 'Proceedings of the 3rd International Symposium on Intelligent Control', pp. 382–384.

Borenstein, J. and Koren, Y. (1989), 'Real-time obstacle avoidance for fast mobile robots', *IEEE Transactions on systems, Man, and Cybernetics* **19**(5), 1179–1187.

Borenstein, J. and Koren, Y. (1990), Real-time obstacle avoidance for fast mobile robots in cluttered environments, *in* 'Proceedings., IEEE International Conference on Robotics and Automation', IEEE, pp. 572–577.

Borenstein, J., Koren, Y. et al. (1991), 'The vector field histogram-fast obstacle avoidance for mobile robots', *IEEE transactions on robotics and automation* **7**(3), 278–288.

Buniyamin, N., Ngah, W. W., Sariff, N., Mohamad, Z. et al. (2011), 'A simple local path planning algorithm for autonomous mobile robots', *International journal of systems applications, Engineering & development* **5**(2), 151–159.

Elfes, A. (1986), A sonar-based mapping and navigation system, *in* 'Proceedings. 1986 IEEE International Conference on Robotics and Automation', Vol. 3, IEEE, pp. 1151–1156.

Giesbrecht, J. (2004), Global path planning for unmanned ground vehicles, Technical report, DEFENCE RESEARCH AND DEVELOPMENT SUFFIELD (ALBERTA).

Khaksar, W., Vivekananthen, S., Saharia, K. S. M., Yousefi, M. and Ismail, F. B. (2015), A review on mobile robots motion path planning in unknown environments, *in* '2015 IEEE International Symposium on Robotics and Intelligent Sensors (IRIS)', IEEE, pp. 295–300.

Khatib, O. (1985), Real-time obstacle avoidance for manipulators and mobile robots, *in* 'Proceedings. 1985 IEEE International Conference on Robotics and Automation', Vol. 2, IEEE, pp. 500–505.

Koren, Y., Borenstein, J. et al. (1991), Potential field methods and their inherent limitations for mobile robot navigation., *in* 'ICRA', Vol. 2, pp. 1398–1404.

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

Moravec, H. and Elfes, A. (1985), High resolution maps from wide angle sonar, *in* 'Proceedings. 1985 IEEE international conference on robotics and automation', Vol. 2, IEEE, pp. 116–121.

Moravec, H. P. (1987), Certainty grids for mobile robots, *in* 'Jet Propulsion Lab., California Inst. of Tech., Proceedings of the Workshop on Space Telerobotics, Volume 1'.

Oroko, J. A. and Nyakoe, G. (2022), Obstacle avoidance and path planning schemes for autonomous navigation of a mobile robot: a review, *in* 'Proceedings of the Sustainable Research and Innovation Conference', pp. 314–318.

Palossi, D., Conti, F. and Benini, L. (2019), An open source and open hardware deep learning-powered visual navigation engine for autonomous nano-uavs, *in* '2019 15th International Conference on Distributed Computing in Sensor Systems (DCOSS)', IEEE, pp. 604–611.

Raja, P. and Pugazhenthi, S. (2012), 'Optimal path planning of mobile robots: A review', *International journal of physical sciences* **7**(9), 1314–1320.

Sabudin, E., Omar, R., CKAN, H. and Melor, C. K. (2016), 'Potential field methods and their inherent approaches for path planning', *ARPN J. Eng. Appl. Sci* **11**(18), 10801–10805.

Saravanakumar, S. and Asokan, T. (2013), 'Multipoint potential field method for path planning of autonomous underwater vehicles in 3d space', *Intelligent Service Robotics* **6**(4), 211–224.

Zhang, H.-y., Lin, W.-m. and Chen, A.-x. (2018), 'Path planning for the mobile robot: A review', *Symmetry* **10**(10), 450.