DEGREE PROJECT IN COMPUTER SCIENCE AND ENGINEERING, SECOND CYCLE, 30 CREDITS

STOCKHOLM, SWEDEN 2016

Obstacle avoidance for platforms in three-dimensional environments

JOHAN EKSTRÖM

KTH ROYAL INSTITUTE OF TECHNOLOGY
SCHOOL OF COMPUTER SCIENCE AND COMMUNICATION

KTH Computer Science
and Communication

Obstacle avoidance for platforms in three-dimensional environments

JOHAN EKSTRÖM

Master's Thesis at CVAP
Supervisor: Patric Jensfelt
Examiner: Joakim Gustafson

Abstract

The field of obstacle avoidance is a well-researched area. Despite this, research on obstacle avoidance in three dimensions is surprisingly sparse. For platforms which are able to navigate three-dimensional space, such as multirotor UAVs, such methods will become more common.

In this thesis, an obstacle avoidance method, intended for a three-dimensional environment, is presented. First the method reduces the dimensionality of the three-dimensional world into two dimensions by projecting obstacle observations onto a two-dimensional spherical depth map, retaining information on direction and distance to obstacles. Next, the method accounts for the dimensions of the platform by applying a post-processing on the depth map. Finally, knowing the motion model, a look-ahead verification step is taken, using information from the depth map, to ensure that the platform does not collide with any obstacles by not allowing control inputs which leads to collisions. If there are multiple control input candidates after verification that lead to velocity vectors close to a desired velocity vector, a heuristic cost function is used to select one single control input, where the similarity in direction and magnitude of the resulting and desired velocity vector is valued.

Evaluation of the method reveals that platforms are able to maintain distances to obstacles. However, more work is suggested in order to improve the reliability of the method and to perform a real world evaluation.

Referat

Kollisionsundvikande metoder för plattformar i tredimensionella miljöer

Fältet inom kollisionsundvikande är ett välforskat område. Trots detta så är forskning inom kollisionsundvikande metoder i tre dimensioner förvånansvärt magert. För plattformar som kan navigera det tredimensionella rummet, såsom multirotor-baserade drönare kommer sådana metoder att bli mer vanliga.

I denna tes presenteras en kollisionsundvikande metod, menad för det tredimensionella rummet. Först reduceras dimensionaliteten av det tredimensionella rummet genom att projicera hinderobservationer på ett tvådimensionellt sfärisk ark i form av en djupkarta som bibehåller information om riktning och avstånd till hinder. Därefter beaktas plattformens dimensioner genom att tillämpa ett efterbehandlingssteg på djupkartan. Till sist, med kunskap om rörelsemodellen, ett verifieringssteg där information från djupkartan används för att försäkra sig om att plattformen inte kolliderar med några hinder genom att inte tillåta kontrollinmatningar som leder till kollisioner. Om det finns flera kontrollinmatningskandidater efter verifikationssteget som leder till hastighetsvektorer nära en önskad hastighetsvektor så används en heuristisk kostnadsfunktion, där likheten i riktning och magnitud av den resulterande vektorn och önskade hastighetsvektorn värderas, för att välja en av dem.

Utvärdering av metoden visar att plattformar kan bibehålla avstånd till hinder. Dock föreslås ytterligare arbete för att förbättra tillförlitligheten av metoden samt att utvärdera metoden i den verkliga världen.

Contents

Chapter 1

Introduction

Automatization and the application of robotics are changing the way the industry works. The impact it has can be compared to that of the introduction of the assembly line in the beginning of the 20th century. In factories, robots are used for dangerous, precise and repetitive tasks and as such have a natural place at the assembly line. At home, there are robots that vacuum and mow the lawn. Robots are entering the social sphere at an ever accelerating rate.

One aspect of the introduction of robots in the industry is the employment of *Unmanned Aerial Vehicles*, or UAVs for short. UAVs have found a wide range of uses, such as delivering packages, performing search and rescue missions for law enforcements and providing surveillance for intelligence agencies. There are a number of commercially available consumer UAVs, making UAVs a product for hobbyists as well. Photographers and film makers can use relatively cheap UAVs to capture footage from the air with unprecedented mobility without the need for expensive and cumbersome cranes.

Typically, UAVs excel in large scale mapping missions, where they, traversing over a large area of land, use sensor data to build highly detailed maps. Usually, this is a task that a fixed-wing UAV, flying the same way planes do, is able to perform amicably, because it is often not the case that a UAV has to take various environmental obstacles into account, given that the UAV is flying at relatively high altitudes. However, there is interest in having UAVs to perform similar mapping tasks in more local environments, such as in residential areas. This brings the UAV closer to obstacles and these have to be taken into account. As with every other vehicle, a collision could result in great property and personal damages and therefore have to be avoided to the best of ability.

A fixed-wing UAV may not be the best fit for these types of missions. A more appropriate UAV type is the multirotor UAV, using the same principles as the helicopter. Skilled multirotor UAV operators can easily navigate a UAV in an urban environment. However, there is also interest in allowing a computer to operate UAVs, thereby automating the process of navigation. This requires control algorithms to take potential obstacle encounters into account, especially if the environment is

completely unknown. There is also interest in taking this further by having UAVs fly around autonomously indoors. One example is that UAVs were used in order to inspect structural damages to buildings affected by the hurricane Katrina[1].

1.1 Problem Statement

This thesis regard the problem of avoiding obstacles in a three-dimensional, GPS-denied, static, unknown, cluttered environment with a multirotor UAV. The scenario is that a human operator, or an autonomous global path planner, sends a command input with the purpose of reaching a desired velocity. As the multirotor traverses the environment, obstacles may appear, and depending on the command input, it may collide with obstacles. The problem then is to not allow command inputs which may lead to collisions and/or suggest command inputs similar to the desired one which can avoid collisions. This thesis will attempt to implement an obstacle avoidance method and in the end find out if the method is a reasonable approach to solve this problem.

1.2 Outline

The purpose of this thesis is to provide an understanding of collision avoidance techniques for actors in three-dimensional environments with six degrees of freedom and the problems that may arise as a result, as well as to provide a reactive collision avoidance algorithm. In the background section, various reactive collision avoidance strategies are presented, including some collision avoidance strategies developed by others. In the method section, a collision avoidance algorithm is provided based on the methods found in the background section as well as strategies for how sensor data could be used with the algorithm. In the results section, the behaviour of the collision avoidance algorithm is analysed. The thesis ends with a discussion section, where we discuss how the method compares to other methods, strengths and drawbacks of the collision avoidance algorithm, how the algorithm could be extended and potential future research.

Chapter 2

Background

In this chapter, we will study a selection of works regarding obstacle avoidance and background information needed to understand and motivate the method developed in this thesis.

Most collision avoidance techniques have been developed for actors in two-dimensional environments. However, many of these techniques can be extended to three dimensions. In this chapter, we present various collision avoidance approaches, their strengths and drawbacks and various approaches to UAV collision avoidance provided by others.

2.1 Multirotor motion model

Multirotor UAVs are platforms which are able to move by pushing air away from it using a set of rotors. Designs of multirotor UAVs can vary greatly, but a common configuration is to place the rotors symmetrically with respect to the horizontal plane.

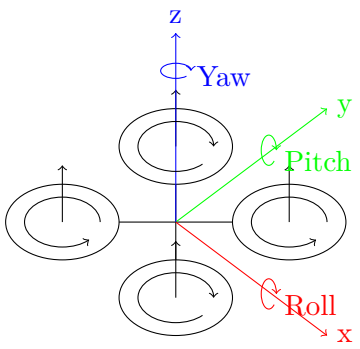


Figure 2.1. A diagram of a multirotor UAV with four rotors and the relations between the coordinate system and pitch, roll and yaw. The axis which roll is rotates around denotes the heading.

A diagram of a quadrotor UAV platform can be seen in Figure 2.1. Each rotor produces a thrust directed along the normal of the rotor, as well as torque. Each pair of contralateral rotors rotate in different directions in order to be able to reliably control the torque forces. By changing the individual rotor speeds, the multirotor is able to move and rotate in all directions.

However, the motion model is extremely complex. As shown in[2, 3, 4] multirotor platforms are under-actuated and are dynamically unstable and estimating the translational and rotational velocities of them is a nonlinear problem and requires careful consideration how a wide variety of parameters may affect the system. The motion dynamics depend not only on the rotor speeds but also on factors such as the mass of the multirotor, inertia, the pose, air pressure, drag, wind, etc. Due to the complexity of modelling multirotor UAVs, approximations are often used.

2.1.1 Frames of reference

Before defining multirotor UAV motion models, it is important to detail the relevant frames of references. Let $\{A\}$ represent a right-handed inertial frame of reference, where vectors $\{\overrightarrow{a_1}, \overrightarrow{a_2}, \overrightarrow{a_3}\}$ are unit vectors corresponding to the coordinate axis defined by $\{\overrightarrow{x}, \overrightarrow{y}, \overrightarrow{z}\}$. Let B be the body frame of reference with unit vectors $\{\overrightarrow{b_1}, \overrightarrow{b_2}, \overrightarrow{b_3}\}$. The orientation of the body frame can be found with a rotation matrix R_B, where $\overrightarrow{b_1} = R_B \overrightarrow{x}$, $\overrightarrow{b_2} = R_B \overrightarrow{y}$, $\overrightarrow{b_3} = R_B \overrightarrow{z}$. It is defined by the roll (ϕ), pitch (θ) and yaw (ψ) angles[5]:

$$R_B = \begin{pmatrix} \cos\psi\cos\theta - \sin\phi\sin\psi\sin\theta & -\cos\phi\sin\psi & \cos\psi\sin\theta + \cos\theta\sin\phi\sin\psi \\ \cos\theta\sin\psi + \cos\psi\sin\phi\sin\theta & \cos\phi\cos\psi & \sin\psi\sin\theta - \cos\psi\cos\theta\sin\phi \\ -\cos\phi\sin\theta & \sin\phi & \cos\phi\cos\theta \end{pmatrix}$$

For future reference, we introduce the body-plane fixed frame $\{C\}$, which corresponds to the heading of the platform in relation to the horizontal plane of $\{A\}$, with unit vectors $\overrightarrow{c_1} = R_C \overrightarrow{x}, \overrightarrow{c_2} = R_C \overrightarrow{y}, \overrightarrow{c_3} = \overrightarrow{z}$. The rotation matrix R_C only depends on the yaw of the platform:

$$R_C = \begin{pmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

2.1.2 Kinematic models

A typical model of the UAV kinematics is as follows[3, 5]:

$$\dot{\zeta} = v \tag{2.1}$$

$$m\dot{v} = mg\overrightarrow{a_3} + R_B F \tag{2.2}$$

$$\dot{R_B} = R_B \omega_\times \tag{2.3}$$

$$I\dot{\omega} = -\omega \times I\omega + \tau \tag{2.4}$$

where $\zeta = [x, y, z]^T$ is the position of the multirotor, $v = [\dot{x}, \dot{y}, \dot{z}]^T$ is the velocity, $\omega = [\dot{\phi}, \dot{\theta}, \dot{\psi}]^T$ is the angular velocity of the multirotor and $\omega_\times$ is the skew symmetric matrix of ω, I is the inertia matrix and $\tau = [\tau_1, \tau_2, \tau_3]^T$ is the torque generated by the rotors.

Typically, the state space representation of a multirotor is as follows[3, 4, 6]

$$X = \begin{bmatrix} x & y & z & \dot{x} & \dot{y} & \dot{z} & \phi & \theta & \psi & \dot{\phi} & \dot{\theta} & \dot{\psi} \end{bmatrix}^T$$

where x, y, z is the current position, $\dot{x}, \dot{y}, \dot{z}$ are the linear velocities, ϕ, θ, ψ is roll, pitch and yaw respectively and $\dot{\phi}, \dot{\theta}, \dot{\psi}$ are the angular velocities. A reasonable approach for finding the next state is finding expressions for $\ddot{x}, \ddot{y}, \ddot{z}, \ddot{\phi}, \ddot{\theta}, \ddot{\psi}$ which depend on some control input[3, 4, 6, 5]. With the state space representation, we can represent the change of a state given some control input U as $\dot{X} = f(X, U)$.

The important consequence of the motion model is that the linear accelerations in the horizontal plane is a function of the roll, pitch and yaw of the multirotor. A change in pitch alters the acceleration along c_1 and a change in roll alters the acceleration along c_2. A change in overall rotor thrust results in an altered upward acceleration (along z). By extension, this means that the kinematic dynamics of the multirotor can be expressed by the overall thrust of the rotors and the torque[5]. Most importantly, the force vector F can be described as follows:

$$F = T_\Sigma \overrightarrow{z} + \Delta \tag{2.5}$$

where Δ is used to model certain aerodynamical phenomena when the multirotor is not in hover, such as drag and rotor flapping. Also note that F is expressed in body-fixed frame B.

Implementing a motion model for a specific multirotor type and defining how the system translates some control command into rotor speeds to best carry out an action is an arduous task and beyond the scope of this thesis. Instead, for this thesis we assume that there is an existing control system which allows for a phenomenological approximation, and treat the motion model as a dynamic point model.

It is important to note that the following model does not aim to emulate any known multirotor type or control system. The purpose is to define a motion model which can be used for the purpose of evaluating the obstacle avoidance method rather than evaluating the accuracy of the motion model.

We model the linear velocity of the multirotor based on the motion model found in [7], which is described as follows:

$$\dot{\zeta} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} = v \tag{2.6}$$

$$\ddot{v} = \begin{bmatrix} \frac{1}{\tau_x}(-\ddot{x} + \dot{v}_{x,C}) \\ \frac{1}{\tau_y}(-\ddot{y} + \dot{v}_{y,C}) \\ \frac{1}{\tau_z}(-\ddot{z} + \dot{v}_{z,C}) \end{bmatrix} \tag{2.7}$$

where τ_x, τ_y, τ_z are time constants and $\dot{v}_{x,C}, \dot{v}_{y,C}, \dot{v}_{z,C}$ are the control accelerations to be found.

For future reference, $\ddot{v}$ can be rewritten as follows:

$$\ddot{v} = M_\tau(-\dot{v} + \dot{v}_C)$$

where:

$$M_\tau = \begin{bmatrix} \frac{1}{\tau_x} & 0 & 0) \\ 0 & \frac{1}{\tau_y} & 0 \\ 0 & 0 & \frac{1}{\tau_z} \end{bmatrix}$$

As previously mentioned, the linear velocity depends on the pose. However, given an observation of the acceleration, we can deduce an approximation of the actual angles by using the relationship between the acceleration and the force vector in equation 2.2. Combining equation 2.2 and 2.5, we get the following expression:

$$T_\Sigma \overrightarrow{b_3} = m\dot{v} - mg\overrightarrow{a_3} - R_B\Delta \tag{2.8}$$

where Δ is modelled as the drag induced on the multirotor:

$$R_B\Delta = Dv^2 \tag{2.9}$$

where D is a drag matrix.

Since $\overrightarrow{b_3}$ is a unit vector, $\|\overrightarrow{b_3}\| = 1$. In order for the expression to be balanced, $T_\Sigma = \|m\dot{v} - mg\overrightarrow{a_3} - R_B\Delta\|$. By dividing with T_Σ, we get an expression for $\overrightarrow{b_3}$.

The roll and pitch angles are found as follows:

$$\phi = \frac{\pi}{2} - \arccos(\overrightarrow{b_3} \cdot \overrightarrow{c_1}) \tag{2.10}$$

$$\theta = \frac{\pi}{2} - \arccos(\overrightarrow{b_3} \cdot \overrightarrow{c_2}) \tag{2.11}$$

which is possible to assess since $\overrightarrow{c}$ only depends on the yaw, which can be modelled independently.

Observant readers may note that if the multirotor is falling at terminal speed, $T_\Sigma = 0$, which means that the number of solutions are infinite. Therefore we disallow velocities which would lead to such a case.

Since the yaw of the multirotor has a marginal influence on the linear speed and angular velocities (specifically, yaw only determines the direction of the linear speeds, roll and pitch), the yaw is modelled separately:

$$\ddot{\psi} = \frac{1}{\tau_\psi}(-\dot{\psi} + \ddot{\psi_C}) \tag{2.12}$$

We define a control input U as follows:

$$U = \begin{bmatrix} \dot{v}_C & \ddot{\psi}_C \end{bmatrix}^T \tag{2.13}$$

For this motion model, while it is possible to deduce the pitch and roll angles from the acceleration, it does not result in less parameters for the state space representation, but instead the pitch/roll information is replaced with acceleration information. The state space X and $\dot{X}$ can be defined as follows:

$$X = \begin{pmatrix} x \\ y \\ z \\ \dot{x} \\ \dot{y} \\ \dot{z} \\ \ddot{x} \\ \ddot{y} \\ \ddot{z} \\ \psi \\ \dot{\psi} \\ \ddot{\psi} \end{pmatrix}, \dot{X} = f(X,U) = \begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ \ddot{x} \\ \ddot{y} \\ \ddot{z} \\ \frac{1}{\tau_x}(-\ddot{x} + \dot{v}_{x,C}) \\ \frac{1}{\tau_y}(-\ddot{y} + \dot{v}_{y,C}) \\ \frac{1}{\tau_z}(-\ddot{z} + \dot{v}_{z,C}) \\ \dot{\psi} \\ \ddot{\psi} \\ \frac{1}{\tau_\psi}(-\ddot{\psi} + \ddot{\psi}_C) \end{pmatrix} \tag{2.14}$$

2.1.3 Rudimentary control solution

The purpose of the command system is to find control inputs which allows the multirotor to reach a certain velocity v_{des} and a desired angular yaw velocity $\dot{\psi}_{\text{des}}$. A rudimentary solution to the control problem can be found as a result of assuming that v_{des} is reached by some control input $\dot{v}_{C,\text{opt}}$ with the following relation:

$$v_{\text{des}} = v_0 + \delta\dot{v} = v_0 + \delta(\dot{v}_0 + \delta\ddot{v}) = v + \delta(\dot{v}_0 + \delta(M_\tau(-\dot{v}_0 + \dot{v}_{C,\text{opt}}))) \tag{2.15}$$

where $v_0, \dot{v}_0$ is the current velocity and acceleration and δ is a delta time.

Solving for $\dot{v}_{C,\text{opt}}$, we get the following expression:

$$\dot{v}_{\text{C,opt}} = \frac{1}{\delta} M_\tau^{-1}(\frac{v_{\text{des}} - v_0}{\delta} - \dot{v}_0) + \dot{v}_0 \tag{2.16}$$

However, we do not assume that infinite accelerations are permissible. We assume that the constraints on the control accelerations are as follows:

$$\|\dot{v}_C\| \leq \dot{v}_{\max} \tag{2.17}$$

Similarly, we can find the control torque for the yaw (assuming a desired angular yaw velocity ψ_{des}):

$$\dot{\psi}_{\text{C,opt}} = \frac{\tau_\psi}{\delta}(\frac{\psi_{\text{des}} - \psi}{\delta} - \dot{\psi}) + \dot{\psi} \tag{2.18}$$

with a similar acceleration constraint:

$$\|\dot{\psi}_C\| \leq \dot{\psi}_{\max} \tag{2.19}$$

2.2 Obstacle avoidance methods

In this section, we study a selection of obstacle avoidance methods.

2.2.1 Potential field methods

Artificial potential fields methods are very popular methods used to induce reactive collision avoidance behaviours, much due to the simplicity regarding the assumptions made. Artificial potential fields reduce the problem of navigation to a physics problem. The main theory is that the actor is considered a particle which is affected by various forces. Exactly what these forces are and how they are applied to the actor is essentially arbitrary, but usually obstacles are sources of repelling forces on the particle (more work is required in order to move towards obstacles, meaning higher potential energy) while destinations, or waypoints, are sources of attractive forces (points closer to these positions have less potential energy). The standard way of calculating the forces at a position x, as it was first defined as in [8], is as follows:

$$F(x) = F_{\text{att}}(x) + F_{\text{rep}}(x) \tag{2.20}$$

$$F_{\text{att}}(x) = -k_{\text{att}}(x - x_{\text{des}}) \tag{2.21}$$

$$F_{\text{rep}}(x) = \begin{cases} k_{\text{rep}}(\frac{1}{\rho(x)} - \frac{1}{\rho_0})\frac{1}{\rho^2(x)}\frac{x - x_{\text{obs}}}{\rho(x)} & \text{if } \rho(x) \leq \rho_0 \\ 0 & \text{otherwise} \end{cases} \tag{2.22}$$

where $\rho(x)$ is the distance to the obstacle, ρ_0 is the distance of influence and $k_{\text{att}}, k_{\text{rep}}$ determines the overall strength of the forces.

Many of the early implementations assumed that the environment was known and were used for global path planning. However, adapting potential fields methods to work reactively is simple. In [9] the Virtual Force Field algorithm (VFF) is presented. The VFF was implemented and tested on a mobile differential drive robot using sonar to sense the environment. Using a histogram grid with certainty values (representing the belief that an obstacle is occupying a point in space) in conjunction with sonar data, the authors were able to estimate the relative positions of obstacles. The attractive force was proportional to the distance of a target point and the repelling forces for each cell in the certainty grid were calculated by weighing the force by the inverse cell distance and certainty value. The resulting force was the sum of all forces. In order to adapt the resulting force to be used with the differential drive motion model, the steering rate was set to be proportional to the angle of the resulting force vector and the robot heading. The authors reported superior performance compared to other previous methods such as edge following,

which would for example require the robot to stop at certain points in order to record data.

There are some issues regarding potential fields which are "inherent to this principle", as argued in [10]. Using a mathematical model to represent how the robot dynamics is affected by changes, the authors argue that there are four main problems with potential fields. The first problem is that potential fields are prone to having local minima situations, which potential fields are entirely inadequate at escaping. The second problem is that the method collapses all forces into one singular force, which removes information on obstacle placements, which could make it impossible to traverse narrow passages in some situations, even if it would be physically possible and desirable. The third and fourth problems both relate to the fact that oscillations can occur as a result of moving close to obstacles or through narrow corridors. The findings of these problems prompted the authors to abandon potential fields and develop a new method called Vector Field Histogram.

2.2.2 Vector Field Histogram (VFH)

The Vector Field Histogram method is a reactive collision avoidance method developed for wheel-based motion models in a two-dimensional environment as it was first introduced in [11] and expanded upon in [12].

In the method proposal, the setup is very similar to the one in [9], where a differential drive robot equipped with sonar sensors updates a histogram grid with certainty values. For each time step, *an active window*, a subset of the histogram grid with the robot position as center, is defined. The most important aspect of this method is a data reduction step that takes the active window and transforms it into a polar histogram. Each histogram produced by this reduction defines sectors with discrete angles representing their heading around the robot with *polar obstacle density* values, a metric which collapses the belief of obstacles occupying cells and the distances to them in a certain direction (higher values mean more probable, closer obstacles).

The polar histogram thus represents the belief about obstacles in certain directions. The assumption is that directions with polar densities higher than a certain threshold are not permissible to move towards. Therefore, any steering command that does not move the robot towards these high density polar directions are less likely of colliding with obstacles. The remaining low-density sectors are considered permissible directions to head towards. A sensible steering command could then be to move towards the direction with the smallest angular difference to the heading of the robot and the target destination.

In [13], extensions to the original VFH method was introduced, called VFH+. The original implementation did not consider the dimensions of the robot, which was rectified by *enlarging* obstacle cells by the radius of the robot. Also, instead of using the polar density histogram, which could have a tendency of changing rapidly, to decide the steering command directly, a binary polar histogram was developed which is a representation of the polar density histogram where the result

is either 1 or 0 depending on whether if the value of the polar density histogram falls over or under some threshold. The motion model of the robot was also taken into consideration, where vectors which would result in a collision within the robots minimum steering angle (determined as obstacles overlapping with a circle with the constant minimum steering radius) were removed as permissible candidates. Given both the binary histogram and removing vector candidates based on motion model dynamics, a masked histogram is defined, where *valleys*, which is an interval where the values of the histogram is 0, represent permissible directions of travel. While the original implementation left the decision of the steering command ambiguous, the authors provide some methods of finding suitable candidate directions. For each valley, if it is considered *narrow*, the robot should move dead center through it, otherwise, the robot should either move towards the left or right side, or if possible directly towards the goal direction. A heuristic cost function is used in order to determine the best direction.

In [14], additional extensions are provided, called VFH*. While VFH+ only considers the current state of directions when selecting a steering command, VFH* also takes into consideration future configurations. Essentially, the authors use a combination of A* and the heuristic cost function from VFH+ to find the current least cost path. The current direction which leads to the lowest-cost path is selected for the steering command. The benefit of this approach is that it avoids local minimi conditions. Note that this approach requires the configuration space to be available as well as a good motion model.

The main drawback of the VFH method is how it handles the motion dynamics, or rather the lack of it. As mentioned in [15] and [16], VFH does not take into account the vehicle dynamics. Adequate performance of VFH on a non-holonomic robot (meaning a robot which does not have full control over all degrees of freedom at any given time) is therefore not guaranteed. In other words, there may be situations where the robot decides to steer into obstacles at high velocity, since the desire to head towards a given direction does not immediately translate to heading towards that direction instantly. This results in a behaviour where the robot could steer towards a permissible direction, oblivious to the fact that the resulting trajectory would result in a collision. The assumption that has to be made regarding the motion model is that the vehicle can instantly move towards the desired direction. Since this assumption is not necessarily applicable, dynamic motion models have to be considered. Even though VFH+ introduced a method by which to eliminate some vectors by considering the motion model, it relies on the fact that the minimum steering radius remains constant for all velocities. This assumption breaks down for differential drive motion models, to give one example.

Additionally, VFH depends on empirically determined parameters which may vary depending on the environmental context. It is also not obvious if a robot using VFH is able to approach a destination if the path moves the robot towards an obstacle. Since the polar density metric collapses potentially relevant information such as distance to obstacles, determining if a position is in front of or behind an obstacle is unreliable (although such a check could be done using the two-dimensional

histogram grid instead of the polar histogram).

2.2.3 Nearness Diagram (ND)

The Nearness Diagram method, first presented in [17], expanded upon in [18], is similar to the Vector Field Histogram. Both methods use a diagram representation of the environment based on sectors around a robot. They differ mainly in the measured metrics and obstacle avoidance strategy. A Nearness Diagram implementation assumes accurate directional distance measures are recorded at each time step. This is used in order to calculate a *nearness metric* associated with the sectors. The diagram is then searched for *gaps*, denoted as discontinuities in the diagram and a suitable valley is selected as the *free walking area*, which the robot will attempt to move towards. The method to avoid obstacles depends on five identifiable states, which are related to if the robot is too close to obstacles on one side of the free walking area or both; if the valley of the free walking area is *wide* or *narrow*; and if the goal is located in the free walking area or not.

This approach was designed primarily for holonomic motion models in a two-dimensional environment and requires the use of accurate omnidirectional range finders such as 2D laser scanners. As a result, it is not obvious how the method could be used in a three-dimensional setting, especially since the proposed decision tree is only applicable for the two-dimensional environment.

2.2.4 Obstacle-Restriction Method (ORM)

The obstacle-restriction method, presented in [19], is an obstacle avoidance method which uses information on obstacle positions directly, as opposed to the previous methods where information is collapsed to some capacity. The environment is thus represented as a two-dimensional point cloud, where the points denote obstacle locations.

The ORM has two components: first, a goal (or sub-goal) is selected; second, the method restricts the motion of the robot by finding undesirable directions as a function of the obstacles. The set of desirable directions is then the complement of the set of undesirable directions.

If the desired goal cannot be reached, then a list of candidate sub-goals are introduced, where they are points either between obstacles at a distance longer than the diameter of the robot or in the direction of obstacles, also at a distance longer than the robot diameter. The closest, reachable sub-goal is then chosen.

Undesirable directions are found for each obstacle as the union of two subsets: one representing directions unsuitable for avoidance, the other an "exclusion area" around the obstacle, which is an area defined by the angles encapsulating the obstacle, enlarged by the size of the robot's radius and a safety distance. The union of all sets per obstacle is the set of undesirable directions.

The method was designed for holonomic motion models in a two-dimensional environment. However, there are examples of ORM implementations in a three-

dimensional context[20].

2.2.5 Dynamic Window Approaches

The idea of taking the motion model of the robot into account directly as a means of reactive collision avoidance has taken many forms. Some tangentially relevant methods are the Steer Angle Field[15] and the Curvature-velocity method[21]. Both these methods avoid obstacles by taking the differential drive trajectories into account.

In [16], the dynamic window approach was introduced. It is able to find steering commands which moves the robot towards a target while taking the limited ability of the robot to affect its velocity, defined as a tuple of linear and angular velocities (v, w), into consideration.

The main theory of the method is to create a set of velocities which do not result in collisions and can be expected to be reached in the next time step. First, the set of velocities which do not result in a collision is defined:

$$V_a = \{(v, w) | v \leq \sqrt{2 \cdot \text{dist}(v, w) \cdot \dot{v_b}} \wedge w \leq \sqrt{2 \cdot \text{dist}(v, w) \cdot \dot{w_b}}\} \tag{2.23}$$

where $\dot{v_b}, \dot{w_b}$ are the brake accelerations and $\text{dist}(v, w)$ denotes the smallest distance to an obstacle that intersects the trajectory produced by selecting the velocity (v, w). Next, a set of reachable velocities is defined:

$$V_d = \{(v, w) | v \in [v_a - \dot{v}t, v_a + \dot{v}t] \wedge w \in [w_a - \dot{w}t, w_a + \dot{w}t]\} \tag{2.24}$$

where (v_a, w_a) is the actual velocity. This lets us define a set with reachable velocities which do not result in collision:

$$V_r = V_s \cap V_a \cap V_d \tag{2.25}$$

where V_s is the set of all possible velocities. Given a destination or direction of travel, the following heuristic cost function is used in order to find the most appropriate velocity:

$$\text{G}(v, w) = \alpha \text{heading}(v, w) \cdot \beta \text{dist}(v, w) \cdot \gamma \text{velocity}(v, w) \tag{2.26}$$

where $\text{heading}(v, w)$ denotes the alignment between the robot heading and the direction of the destination/direction of travel and $\text{velocity}(v, w)$ the projection of v.

The authors used this principle to invoke a reactive collision avoidance behaviour on a synchronous drive motion model robot. The trajectories were approximated as circular arcs. They reported great results and high speeds and argued that the results were comparable to other methods.

In [22], the authors present two additions to the dynamic window approach, one being the holonomic dynamic window approach (the other one how to implement the dynamic window approach for global navigation). Given a holonomic motion model,

it demonstrates how the velocity and trajectory is computed in the two-dimensional case.

2.3 Related work on obstacle avoidance for multirotor UAVs

Much of the research regarding multirotor obstacle avoidance tend to be outdoors where there is usually more space to manoeuvre. One area of research which is closely related to the thesis is collision avoidance in urban canyons, where obstacles tend to be far apart and do not tend to be complex in height, and essentially reduce the problem to a two-dimensional collision avoidance problem. Even in the cases where the authors use simulations, most authors assume that the altitude of the multirotor is maintained.

A report regarding UAV missions to inspect damaged structured in the aftermath of Katrina detail some guidelines when carrying out such missions[1]. Some of the findings include the ability to approach an obstacle to a minimum of 2-5m, and that obstacles have to be considered in all directions.

2.3.1 Optical flow methods

One of the most dominant domains of research regarding UAVs is how to apply optical flow techniques. The obvious reason is that optical flow provides a range of useful information regarding both scene geometry and the motion of the platform. The assumption is that relative differences in images, usually represented as the velocity vector of moving pixels, generated by a moving camera arise mainly as a result of translations and rotations of the camera. If a translation and rotation is known, and obstacles have distinguishing textures, then geometric information can be estimated. An important consequence of optical flow is that the magnitude of pixel velocities, normalized by the velocity of the camera, is inversely proportional to the distance of objects, meaning that higher pixel velocity is a consequence of being close to objects. A consequence of this is also that depth estimates are possible; some methods to produce these estimates are presented in [23][24]. A discussion on the applications of optical flow for UAVs is found in [25].

A lot of collision avoidance methods which utilize optical flow work on the principle of steering away from either the left and right side by changing the yaw of the UAV, depending on which side has a higher optical flow (higher flow implies closer obstacles). In [26], one such method was implemented and tested on a simulated UAV using the Autopilot[1] software in three different two-dimensional environments. The UAV was running in these environments for five minutes 100 times. The authors reported that the simulated UAV was able to navigate the environments most of the times, with some cases resulting in collisions. In [27], they expanded upon the original implementation, where they introduced a way of limiting the forward

[1] http://autopilot.sourceforce.net

translation speed by calculating an approximate time-to-contact as well as presenting a method of compensating for optical flow induced by rotation. Using the same test environments and setup, the authors reported that the simulated UAV was able to avoid collision for all simulations.

A similar approach can be found in [28]. They begin by presenting a derivation of the dynamic motion model of a quadrotor UAV as a function of the rotor speeds. The method was tested by simulating a UAV using the virtual reality program in Matlab. Two cases were tested: one case where the UAV had to avoid a pillar; and the other one where the UAV had to traverse down a narrow corridor. The authors reported that the UAV was able to avoid collisions in both environments.

In [29], based on [30], the authors add to this approach. Two fish-eye cameras were set up facing directly left and right and a stereo vision camera facing the front. The stereo vision camera was used in order to detect obstacles in front of the UAV, in which case the UAV turned away from the obstacle. The overall control strategy for the turning was that the stereo vision control strategy took precedence if there was an obstacle in front of the UAV. Otherwise, the control strategy developed for the fish-eye cameras was used. The authors reported that the combination of optical flow and stereo vision was superior to either the stereo vision control strategy or the optical flow strategy on their own.

In [24], the authors present an alternative using a template approach for *gross optical flow process*, where the optical flow image was divided into horizontal and vertical components and the sum of those defined *gross* force vectors which the UAV was subject to, and a *fine optical flow process*, where optical flow was used in combination with an IMU in order to estimate depths, or distances to obstacles, which were then clustered using K-means clustering, which in turn were used in order to create a *fine* force vector. These forces were weighted and summed to produce an output force vector which was used in order to avoid collisions. The method was first evaluated on a dataset in order to analyse the algorithm and was then tested on a real UAV. They reported that the UAV behaved oscillatory which the authors concluded was due to processing delay, but that the UAV was able to avoid collisions.

The main drawback of these methods, as it relates to the motion of the UAV, is that in order to get reliable optical flow values in a given direction, the UAV has to move as perpendicular to that direction as possible. This means that if cameras face left and right, there is not going to be reliable pixel velocity estimates if the UAV is moving towards the left and right. The solutions that the authors present is to not allow these movements, essentially removing two degrees of freedom (y translation and roll), thereby guaranteeing that only forward/backward translations are possible on the horizontal plane.

2.3.2 Potential fields and similar methods

There are a number of examples where potential field methods have been used as obstacle avoidance methods for copter-based UAVS.

In [31], the authors implemented an obstacle avoidance system on a UAV based on a potential fields method. By using a laser scanner and two wide-angle stereo cameras, the UAV was able to sense the environment and build a discrete three-dimensional occupancy grid. The UAV was also discretized into cells, which were subjected to various forces. Because of omnidirectional obstacle detection, the UAV could move in all directions relative to its heading. The attractive force was represented as a waypoint defined by a global path planner. The repulsive force was determined as an average of forces induced on each cell, which was the force induced by the nearest obstacle. The authors implemented a prediction estimate of the UAV trajectory in order to avoid motion state which could cause future collisions. In such cases, the maximum velocity is preemptively lowered. In order to accurately predict the trajectory, they implemented a learned motion model which treated the flight dynamics as a time-discrete linear dynamic system and was optimized based on motion capture data.

In [32], the authors present an obstacle avoidance system using low-cost sensors. They combined distance estimates received from ultrasound and IR sensors in order to benefit from their strengths while being able to avoid their inherent limitations. The estimates were filtered using information from the distance sensors, but also from optical flow and IMU data. This set of sensors were arranged around the UAV as sectors. The estimates retrieved from these sectors were then collapsed into four distance estimates (front, back, left, right). These estimates were then used in a state machine which determined if there was going to be a correction to the pitch and roll of the UAV. The result is that the UAV avoids obstacles by horizontal translation. The authors evaluated the collision avoidance system on a UAV in three test cases: one in which the UAV was in proximity to a corner; one in which the UAV was travelling down a dead end corridor; and the last one in which a person approached the UAV. The results suggest that the UAV was able to avoid collisions in these cases, but the authors note that the corridor case induced an oscillatory motion, but the UAV was nonetheless able to avoid the walls. They evaluated both the collision avoidance implementation, as well as the motion model, using the Gazebo simulator. They reported that the performance of the potential fields implementation was good and that the motion model was accurate.

In [33], the authors managed to implement an obstacle avoidance algorithm on a UAV which reportedly could avoid obstacles at as high speeds as $10m/s$ in an outdoor environment using a Laser Radar (LIDAR). The LIDAR was used in order to estimate the probability of obstacles occupying certain positions. The position for each obstacle was converted to a spherical coordinate system, which was then used with a variant of potential fields. The design of the algorithm allowed the UAV to avoid obstacles in three dimensions, as opposed to the common approach of only avoiding obstacles in two dimensions. However, the limited field of view of the LIDAR merited a reduction in the degrees of freedom of the UAV, namely lateral velocities.

2.3.3 Other methods

In [34], the authors presents an obstacle avoidance method for safe teleoperation of UAVs based on FastSLAM. By using sonars and FastSLAM they estimated the layout of the environment as a two-dimensional map as well as the location of the UAV, and avoided obstacles by restricting the velocity based on the time to collisions to occupied cells. They reported that the UAV was able to avoid obstacles using this method.

In [35], the authors introduce an obstacle avoidance method based on steering away from obstacles when observed. In order to do so, they developed a controller using fuzzy logic, which was optimized using the cross-entropy method. The fuzzy logic controller was later used in [36] and [37], where they utilized monocular SLAM in order to detect obstacles. In both papers, they avoid obstacles by steering away from them.

In [6], the authors present and compare two collision avoidance techniques: Safety-ball and and Mass Point Models. Both methods assume that the UAV is moving towards a destination, and obstacles are avoided by first moving to intermediary points relatively close to the obstacles. The intermediary points for the safety-ball method assumes a safety radius around the UAV, and any intermediary point is placed at that distance from the position of the observation. The Mass Point Model assumes that observations of obstacles are mass points and if there is risk of collision then a sphere is put around the obstacle which the UAV avoids. The authors conclude that the safety-ball model was preferable due to overall superior performance.

In [38], the authors present an obstacle detection system which was used in order to create maps. Using LIDAR and odometry, they built a two-dimensional occupancy grid which was used in order to allow a UAV to follow the walls.

In [39] the authors present an obstacle avoidance algorithm based on the dynamic window approach. They begin by presenting an approximation of the flight dynamics to regard obstacle avoidance in two dimensions. They then present a dynamic window implementation in the context of the resulting motion model. They also introduced a different way of calculating the dynamic cost function in order to remove oscillatory motions which were identified with the standard method of calculating the dynamic cost, and showed that the resulting motion was desirable given the Lyapunov stability criteria. The authors found that implementations using the alternative dynamic cost function performed better than the standard one.

In [7], the authors reduce the problem of obstacle avoidance to a control problem. The authors assume that a UAV is travelling towards a destination and if an obstacle is detected, an intermediate "time-optimal safe point" is found using optimal control methods, which the UAV moves towards. The algorithm was tested in a simulation, where the simulated UAV avoided singular obstacles in order to reach a goal node. In the simulations, the UAV was able to navigate around the obstacles.

In [40], the authors use gaussian process techniques in order to implicitly map the environment with sparse data. With the gaussian process, the authors could

evaluate the probability that a non-discrete three-dimensional point was occupied based on evidence collected from three-dimensional laser scans. Obstacle avoidance was then done by replanning a path to a destination using RRT. The evaluation of the method was done using simulations for three different scenarios: one was to evaluate performance in an urban environment; the other to evaluate performance in a natural environment; and the last as a means to "examine the information-theoretic exploration performance", in other words examining how the map is being built given the observations. In the urban and natural environments, the simulated UAV was able to successfully re-plan the path and avoid obstacles. In the last simulation, the simulated UAV was able to infer the layout of the map successfully.

Chapter 3

Method

In this chapter, we will present a reactive collision avoidance method based on some of the ideas from the literature section. It should be noted that the proposed method is sensor-agnostic, i.e. the idea is not to tailor the method for specific sensors, although some sensors are more suitable for the purpose. Rather, various different sensors can be added to the algorithm after correct preprocessing of the data.

First, we present an outline of the method that is proposed. Then, we present the assumptions made on the environment, and how the environment is ultimately represented to the collision avoidance method. Last, we present how the collision avoidance will work given the representation of the environment.

3.1 Collision avoidance method overview

There is a good reason why most of the collision avoidance algorithms for UAV platforms have been using optical flow methods. The three-dimensional world is complex and an accurate representation would require a lot of data. In a static environment, collision avoidance could be achieved simply by mapping the environment and plan paths around obstacles as they are detected. SLAM methods have been implemented on UAV platforms, but these methods are relatively computationally heavy, making such implementations unfeasable on smaller UAV platforms with weaker hardware. The methods that utilize optical flow, on the other hand, only need to consider local information in order to avoid obstacles. However, the optical flow methods typically restrict the degrees of freedom by only allowing forward translations, which is not necessarily desired.

For the purpose of obstacle avoidance in a cluttered environment, especially an indoor environment, the demand on environment resolution is high. There may be need for centimeter resolutions close to the UAV, but farther obstacles may require less.

This thesis propose a data reduction method for the purpose of obstacle avoidance analogous to VFH, ND and ORM. First, we assume that a set of sensors are

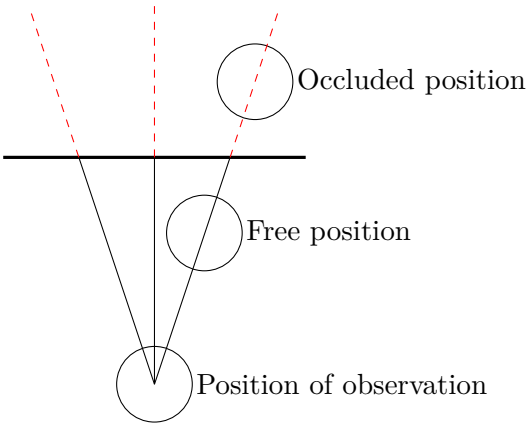


Figure 3.1. Positions on the line between an observation and the UAV (black lines) are regarded as free. Positions on the line behind the observations (red lines) are regarded as occluded.

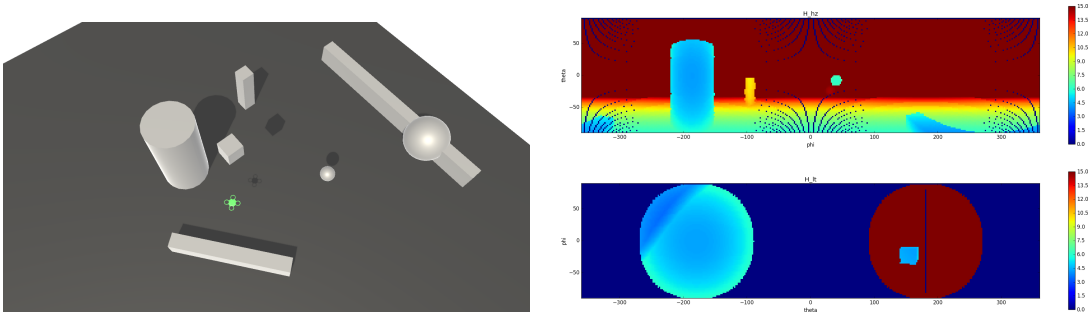
able to produce a high resolution point cloud P, denoting obstacle occupancy. Suitable sensors for this task could be RGB-D cameras, stereo-vision cameras, laser scanners and to a limited extent optical flow. Then we introduce the idea of representing the local environment as a depth map H, which is essentially a depth image projected onto a spherical canvas surrounding the UAV. Each value in H represents the closest distance to an object given a direction, denoted as the polar and azimuthal angles in a spherical coordinate system, from a position of observation, denoted by o.

Once produced, the depth map implicitly defines a volume which describes if a position is either *free* or *occluded*. Free space is defined as a position which is known to not contain an obstacle. Occluded space is defined as the set of positions in which the occupancy is not known. Since the occupancy of a position is not known in occluded space, any obstacle avoidance algorithm which avoids occluded space will avoid collision. The first important assumption is that between observations and the UAV, given that the observation is the closest one for a relative direction in regards to the UAV, there are no obstacles. This means that between observations of obstacles and the UAV is open space in which the UAV is free to roam. The second assumption is that observations occlude the space behind it relative to the UAV.

The process of determining if a position is located in occluded space is straightforward. If:

$$\mathrm{H}(\theta, \phi | o) > r$$

where (r, θ, ϕ) is the spherical coordinate of position p, relative to the observation origo o, is true, the position is in free space, since the the distance to an observation which occludes space in the direction of the position is farther away. As such, it is possible to plan a local path which allows the UAV to avoid obstacles given the depth map.



(a) (b)

Figure 3.2. (a) An example of an environment. Notice the UAV in the middle of the scene. (b) The depth map representation of the local environment, where each pixel represents a direction expressed by ϕ and θ. This snapshot was collected at the position of the UAV.

However, this setup does not consider the dimensions of the UAV. If the UAV could be treated as a simple point then this approach would suffice. However, this is generally not the case. In order to account for the dimensionality of the UAV, we introduce a post-processing step which allows closer distances to "bleed over" to longer distances by enlarging each observation by a safety radius. This allows the depth map to represent distance to obstacles much better than the raw depth map.

The next step is to find a heuristic which ensures that the UAV does not end up in positions in occluded space. The motion model of the UAV is complex, dynamic and potentially volatile, so it should not be neglected. Methods that do not take a motion model into account, such as potential fields (which, additionally, is prone to oscillations), are not considered. Some methods that do consider the motion model is the dynamic window approach and RRT. The following approach is reminiscent of the dynamic window approach.

3.2 Constructing the depth map

First, we note that P can be expressed in different frames (refer to section 2.1.1 for definitions). If P is expressed in A, then a point $p \in P$ corresponds to world coordinates of an observation of an obstacle, which may be preferred if the purpose is to be able to reach a given destination. If P is expressed in C, it is possible to evaluate UAV velocities relative to the UAV heading. However, P will most likely be expressed in B. This is not preferable as roll and pitch information is lost, and since the motion model depend heavily on the pose, these features have to be taken into account anyway. In order to express P in C, it is necessary to be able to estimate the roll and pitch of the UAV, and in order to express P in A, it is also necessary to be able to estimate the yaw. Given that such estimates exist, we can create the rotation matrices R_B and R_C which can be used in order to transform points from one frame to the other. For the remainder of this thesis, we assume

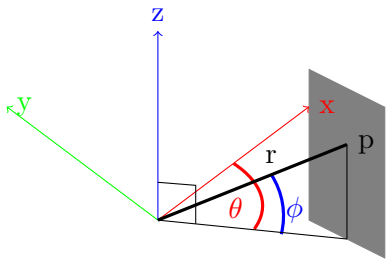


Figure 3.3. The relations between the cartesian coordinate p and spherical coordinates.

that P is expressed in A.

The depth map H is initialized for all polar and azimuthal angles for the position where the observation takes place o in either the inertial frame or body-plane-fixed frame as $\mathrm{H}(\theta, \phi|o) = 0$. An observation is defined as a point $p \in P$ where $p = [p_x, p_y, p_z]^T$, with distance d to the UAV center. For all such observations, we convert the cartesian coordinate to a spherical coordinate:

$$r = d, \theta = \arcsin(\frac{p_z - o_z}{d}), \phi = \arctan2(p_y - o_y, p_x - o_x) \tag{3.1}$$

However, observations where $p_x - o_x = p_y - o_y = 0$ are not defined in H due to $\arctan2(p_y - o_y, p_x - o_x)$ not being defined, and points close to this singularity may suffer floating point errors as a result. A possible solution is to use two different depth maps, $\mathrm{H_{hz}}$ and $\mathrm{H_{lt}}$ where $\mathrm{H_{hz}}$ encodes points with small θ angles and $\mathrm{H_{lt}}$ encodes points with large θ angles.

We redefine H as being a wildcard for these depth maps:

$$\mathrm{H}(p|o) = \begin{cases} \mathrm{H_{hz}}(\theta, \phi) & \text{if } |\theta| < \frac{\pi}{4} \\ \mathrm{H_{lt}}(\theta', \phi') & \text{otherwise} \end{cases} \tag{3.2}$$

where p is the observation, θ, ϕ is found using equation 3.1 and:

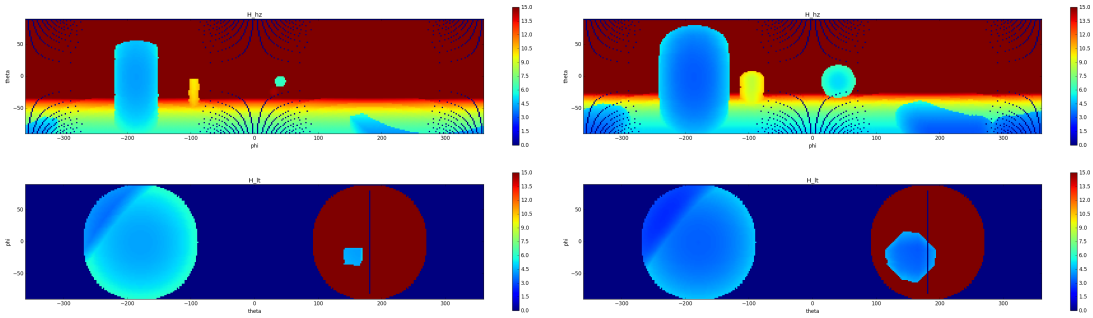
$$\theta' = \arcsin(\frac{p_y - o_y}{d}), \phi' = \arctan2(p_z - o_z, p_x - o_x) \tag{3.3}$$

Assuming that $r \neq 0$, there will be no singularities.

Then we assign values to H for each observation as follows:

$$\mathrm{H}(p|o) \leftarrow \begin{cases} r & \text{if } \mathrm{H}(p|o) = 0 \text{ or } \mathrm{H}(p|o) > r \\ \mathrm{H}(p|o) & \text{otherwise} \end{cases} \tag{3.4}$$

The result is that only the closest obstacles in all directions are used for the obstacle avoidance algorithm. However, the current definition assumes an infinite resolution. Practically, we restrict the resolution of H to $\frac{(\frac{2\pi}{\alpha})^2}{2}$, with α determining the angular



(a) (b)

Figure 3.4. (a) A raw depth map. (b) The depth map after the post-processing step with a safety radius of 1.

width and height of the pixels in the depth map. If, for example, $\alpha = 2$, then each pixel in the depth map roughly correspond to a 2×2 degree area.

3.2.1 Post-processing steps

While the depth map is able to represent the closest distance to an object in a discrete direction, this representation cannot be used for obstacle avoidance purposes if the intention is to be able to maintain a distance to obstacles from the center of the platform. The reason why this is the case is discussed in section 6.2.3. This section will attempt to offer a solution to the problem discussed in the aforementioned section.

We propose a post-processing step of the depth map which inflates the observations (in H) by a safety radius s. Assuming that the UAV occupies an orbital volume, the result is that the depth map is able to take the shape of the UAV into account while only requiring the evaluation of one position. This also provides a convenient solution to potential interpolation and filtering issues, where sensor data might be noisy and/or missing. Assume that $\hat{\mathrm{H}}$ is the post-processed version of H. Assume that all values are initialized as 0.

To demonstrate the enlargement process, we calculate one instance (for one pixel in the depth map). We assume that for a given value of $\mathrm{H}_{\mathrm{hz/lt}}(\theta, \phi) = r$ ($\mathrm{H}_{\mathrm{hz/lt}}$ simply denotes H_{hz} or H_{lt}), there is a spherical boundary with the same radius as the safety radius. We then assume that there is an observation of the horizon of the spherical boundary at distance s from the center of the observation. The angle between the observation of the horizon and the observation of the obstacle is referred to as β with the following relationship (see figure 3.5):

$$\beta = \arctan\left(\frac{s}{r}\right) \tag{3.5}$$

Since the depth map is naturally defined by angles, this means that the enlargement sphere occupies at most a bounding box with angular directions defined as follows:

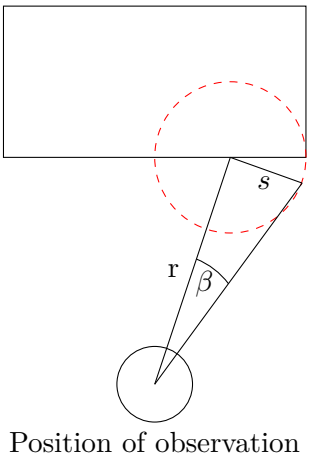


Figure 3.5. The relationship between the observation and the angular enlargement during the post-processing step.

$$\hat{\theta} \in [\theta - \beta, \theta + \beta], \hat{\phi} \in [\phi - \beta, \phi + \beta] \tag{3.6}$$

We assume, however, that the area is circular. This brings the following constraint:

$$(\hat{\theta} - \theta)^2 + (\hat{\phi} - \phi)^2 \leq \beta^2 \tag{3.7}$$

If we assume that $\hat{\theta} \in [\theta - \beta, \theta + \beta]$ we can rewrite the expression so that $\hat{\phi}$ is constrained by β and $\hat{\theta}$:

$$(\hat{\phi} - \phi)^2 \leq \gamma^2 \tag{3.8}$$

where $\gamma^2 = \beta^2 - (\hat{\theta} - \theta)^2$.

For all $\hat{\theta}, \hat{\phi}$ bound by these constraints, we set the values of $\hat{\mathrm{H}}$ as follows:

$$\hat{\mathrm{H}}_{\mathrm{hz/lt}}(\hat{\theta}, \hat{\phi}) = \begin{cases} Y & \text{if } \hat{\mathrm{H}}_{\mathrm{hz/lt}}(\hat{\theta}, \hat{\phi}) = 0 \text{ or } \hat{\mathrm{H}}_{\mathrm{hz/lt}}(\hat{\theta}, \hat{\phi}) > Y \\ \hat{\mathrm{H}}_{\mathrm{hz/lt}}(\hat{\theta}, \hat{\phi}) & \text{otherwise} \end{cases} \tag{3.9}$$

where $\epsilon > 0$ is a small number and:

$$Y = \mathrm{H}_{\mathrm{hz/lt}}(\theta, \phi) - s\cos\Big(\frac{\pi(\hat{\theta} - \theta)}{2\beta}\Big)\cos\Big(\frac{\pi(\hat{\phi} - \phi)}{2\gamma}\Big) \tag{3.10}$$

is the distance after enlargement.

There are some important exceptions. In the directions for which there is no sensor coverage ($\mathrm{H}_{\mathrm{hz/lt}}(\theta, \phi) = 0$), it may not be desirable to assign any enlargement values, as it will imply that an actual observation in that direction has taken place, which is false. We can also assume that the enlargement is only applicable for directions where $\mathrm{H}_{\mathrm{hz/lt}}(\theta, \phi) > 0$, and enlarging distant observations may not be

necessary due to the diminishing size of the circular area and the fact that distant objects pose a lesser risk of collision.

An example of the results of post-processing can be seen in 3.4. After the enlargement process, $\hat{\mathrm{H}}$ is a depth map which takes into account the shape of the UAV.

3.3 Obstacle avoidance approach

Given the depth map $\hat{H}$, we have a method for evaluating if a position is free or occluded while accounting for the shape of the UAV. The next step is to utilize this concept in order to estimate if a control input could result in the UAV entering occluded space. We propose two criteria, evaluated in two phases, which have to be met in order for a control input to be regarded as "permissible" or be rejected.

The first criteria is that the UAV is able to sustain a given control input for a duration of time, hereby referred to as "burn time" b, without entering occluded space. The second criteria is that after the "burn", the UAV is able to stop to a velocity of zero before entering occluded space. If these criteria are met, the control input is assumed to be permissible, otherwise rejected.

In order to evaluate if these criteria are met, an estimate of the trajectory given a control input of the UAV has to be made. The method proposed in this thesis is a search in future state spaces. Generally, given the current state X_0, the next states for a control input U can be estimated using induction as follows:

$$X_{i+dt} = X_i + f(X_i, U)dt \tag{3.11}$$

where dt is the delta time between states.

During the "burn phase", we find the states ranging from X_{dt} to X_b. For each of these states we extract the positions ζ_i (with spherical coordinates (r_i, θ_i, ϕ_i) from origo o) and if:

$$\hat{\mathrm{H}}(\zeta_i|o) > r_i + d \tag{3.12}$$

where $d \geq 0$ is an additional optional safety distance, is true for all i, then at no point (as far as the algorithm is concerned) is the UAV in occluded space during the burn time and the first criteria is met.

During the "brake phase", we use the rudimentary control solution for $v_{\text{des}} = 0$ as control input for each state which results in the UAV stopping reasonably fast. Similarly, beginning from X_b, we find future states up to the last state X_e where $\|v_e\|$ is, or is close to, zero, and evaluate if these states occupy occluded space.

Using this method, it is possible to ascertain if the rudimentary control solution is a permissible control input. However, it may be preferable to find an alternative control input which is close to the rudimentary solution but does not result in collision. One alternative is to do an exhaustive search of control inputs or a focused search around the rudimentary control solution. In this thesis an exhaustive search for a control input is performed.

Given the number of permissible control input candidates, only one control input can be selected. A control input can be selected by maximizing/minimizing some heuristic cost function. This thesis employs the following heuristic cost function (larger value is better):

$$\mathrm{G}(U) = \mathrm{Sign}(v_{\mathrm{des}} \cdot v_b)(v_{\mathrm{des}} \cdot v_b)^2\Big(1 - \Big(\frac{\|v_{\mathrm{des}}\| - \|v_b\|}{\|v_{\mathrm{des}}\|}\Big)^2\Big) \tag{3.13}$$

where v_b is the estimated velocity after the burn time. The reasoning behind this heuristic cost function is that control inputs which result in velocities in the direction of the desired velocity and the ability to maintain the velocity magnitude, are favoured.

Chapter 4

Experimental setup

This chapter details the experimental setup.

4.1 Testing environment

Simulation was done using the Unity game engine[1]. The testing environments were constructed with simple primitive shapes. All code was written in C#.

4.1.1 Platform simulation

The simulated platform for these tests was of a multirotor UAV, which used the motion model defined in section 2.1.2.

4.1.2 Sensor simulation

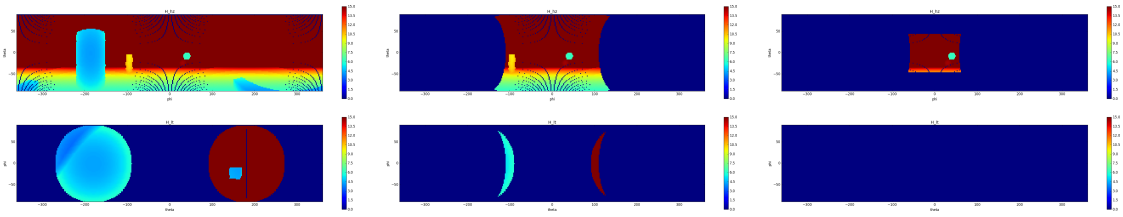
The obstacle avoidance method proposed in this thesis depend on the generation of a point cloud. Suitable methods for generating this point cloud is by using stereo vision and range-finders. The method by which the point cloud is produced is by simulating rangefinders, where measurements are done using active IR/laser sensors. We assume that the surfaces of the obstacles are able to reflect the incoming rays.

The sensor range finding was simulated using Unity's raycasting utilities, which are a part of its physics engine.

Sensor window and range

The assumption is that there is one rigid range-finder with a limited horizontal and lateral FOV placed in the center of the UAV facing forwards. For simplicity, we allow the range-finder to ignore the physical UAV, thereby allowing the sensors to see through it (although in real application this would not be the case). It is assumed that the sensor has a maximum effective range, which may vary depending on the hardware. Distances of observations beyond the effective range are truncated

[1] http://unity3d.com/



(a) (b) (c)

Figure 4.1. The sensor windows affects how the depth map is produced. (a-c) shows the effect of the sensor window with horizontal and lateral FOV of 360/180, 110/110 and 58/45 respectively.

to the maximum effective range. Sensor windows will be denoted as a/b/c, where a is the horizontal FOV, b is the lateral FOV and c is the maximum sensor range.

For real life analogues, the PrimeSense has a horizontal FOV of 57.5° and a lateral FOV of 45°, with a maximum range of 3.4m.[2] The ZED stereo camera developed by Stereo Labs has a reported maximum 110° FOV and a maximum range of 15m.[3] These dimensions are used for simulation.

Omnidirectional FOV (360° horizontal FOV, 180° lateral FOV) is also used for reference.

Noise models

The noise model has two parts: one attempts to model measurement errors and the other one produces missing data.

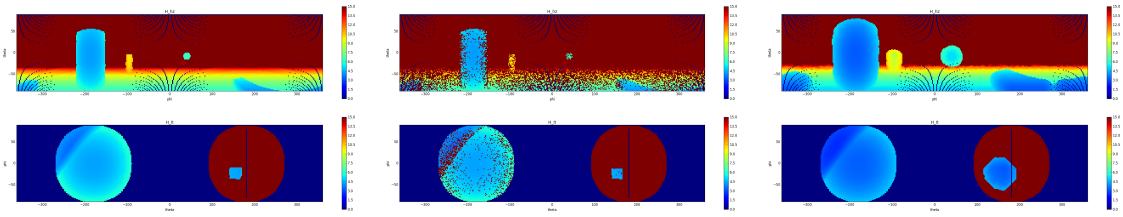
The assumption of the measurement error is based on the error model in [41], where the standard deviation of the measurement error depends on the square of the distance of the measurement. The recorded distance of a measurement d is then sampled from the distribution $d \leftarrow \mathcal{N}(r, \lambda r^4)$, where r is the actual distance and λ is a constant scale factor.

Missing data occurs when an IR/laser ray is not observed to return to the sensor, which can occur if there are no obstacles to reflect the rays or if the angle of incidence Θ is large. Simulating the latter is done by sampling $q \leftarrow \mathrm{U}(a, b)$ where $a \geq 0, a < b, b \leq 1$ and if $q \geq \cos(\Theta)$, we assume that the measurement failed and data is missing. Missing data assumes the value of the maximum effective sensor range.

Update frequency

Updating the depth map at all time steps may not be possible. Instead we assume that there is an interval of time between updates where the UAV can only use an old depth map. The frequency of depth map updates is referred to as the update

[2] http://www.i3du.gr/pdf/primesense.pdf
[3] https://www.stereolabs.com/zed/docs/ZED_Datasheet_2016.pdf



(a) (b) (c)

Figure 4.2. (a) An example of an environment using noise-free sensors. (b) An example of an environment using the sensor noise model. (c) The environment after the enlargement post-processing step. The post-processing step also acts as a filter for missing data.

frequency. At the moment of updating the depth map we assume that data can be gathered instantaneously and observations correspond to the current position.

4.2 Obstacle avoidance evaluation

In order to evaluate the proposed method, we test some cases where the method may work or fail. These are colloquially called *situations*, where given an environment, starting position (and pose) and a goal, the platform should be able to find a path which should allow it to move to a position to the goal without colliding with obstacles. The purpose of testing the obstacle avoidance method in these situations is to assess the ability of the method to overcome some problem such as local minima instances. For example, a typical path planning problem are local minima situations, where finding an actual path to a goal requires the ability to move to position which are objectively worse given some heuristic cost function. A typical situation to demonstrate the ability to overcome local minima is by having an environment with a "U" shape and a goal position which results in the platform moving into the "U" if it moves straight towards the goal (alternatively, the platform has to escape the "U" in order to reach a goal on the other side). However, due to the reactive nature of the proposed method, local minimum conditions are not expected to be avoided. Instead, focus lies on the method's ability to avoid obstacles when a given control input would lead so such a case.

The metric used to assess the ability to avoid obstacles is the closest distance to some obstacle, measured over a number of time points. These values are compared to the safety radius, since if the distance measure is within the safety radius, the UAV occupies occluded space, which is what the method attempts to avoid.

One other metric of interest is the velocity magnitude.

4.2.1 Situation 0: No obstacles

In this situation, there are no obstacles. Moving in any given direction will not result in a collision. The purpose of this situation is not to evaluate the obstacle

Figure 4.3. The environment of situation 0. There are no obstacles.

avoidance, but rather the behaviour of the method given sensory constraints. The obstacle avoidance method only attempts to move to positions which are known to be free, so if there is no sensory coverage of an area then the UAV will be unable to move there. This has some implications for certain types of range-finders with limited FOV such as 2D laser scanners. This fact coupled with pitching and rolling motions and rigid sensors may result in undesirable behaviour.

The UAV will attempt to move forwards, backwards, up, down and right with sensor windows 360/180/15 and 58/45/3.4.

4.2.2 Situation 1: High, wide obstacle

In this situation, the UAV is heading towards a high, wide obstacle. A desirable avoidance should guide the UAV over the obstacle. The UAV may approach the obstacle at different altitudes, which should result in different behaviour. Moving towards the obstacle at low altitude should result in the UAV stopping before it due to sensory constraints. At close to level altitude, the UAV should be able to move over the wall for certain sensors.

The UAV will attempt to move forward with sensor windows 110/110/15 and 58/45/3.4 and also with an additional safety distance of 0m and 1m.

4.2.3 Situation 2: Wall following

In this situation, the UAV is heading towards a wall with a horizontal angle relative to the desired velocity vector. The UAV is expected to be able to move along the

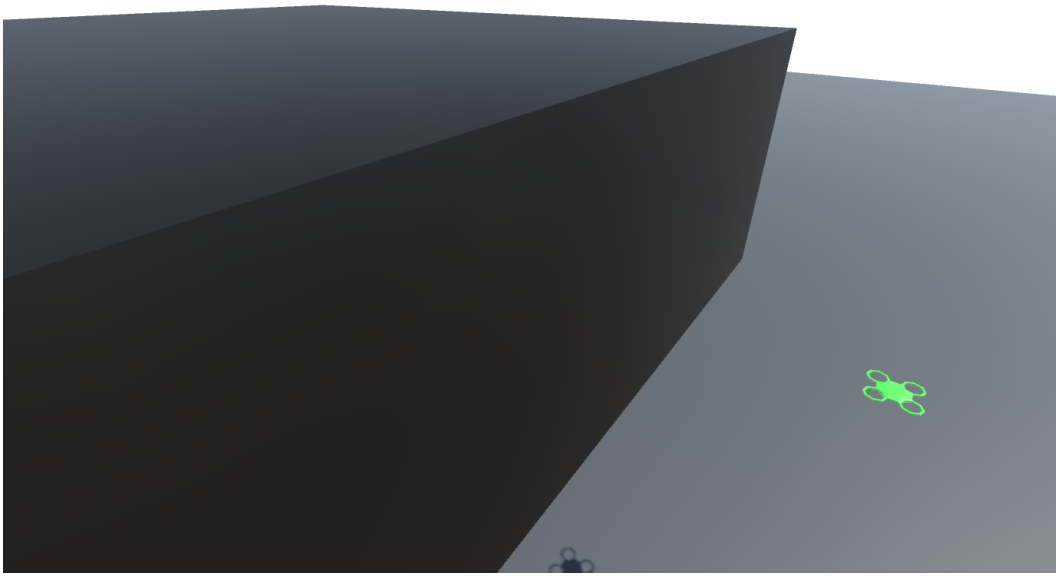


Figure 4.4. The environment of situation 1. The UAV approaches a high wall at varying altitudes. The UAV has to either stop or be able to move over the wall in order to avoid collision.

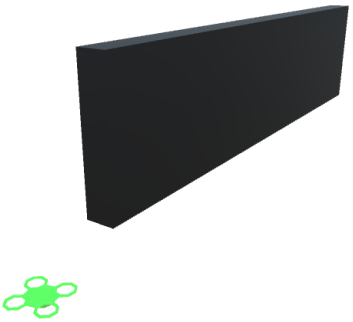


Figure 4.5. The environment of situation 2. The UAV approaches a wall at an angle. The UAV has to either stop or move along the wall in order to be able to avoid collision.

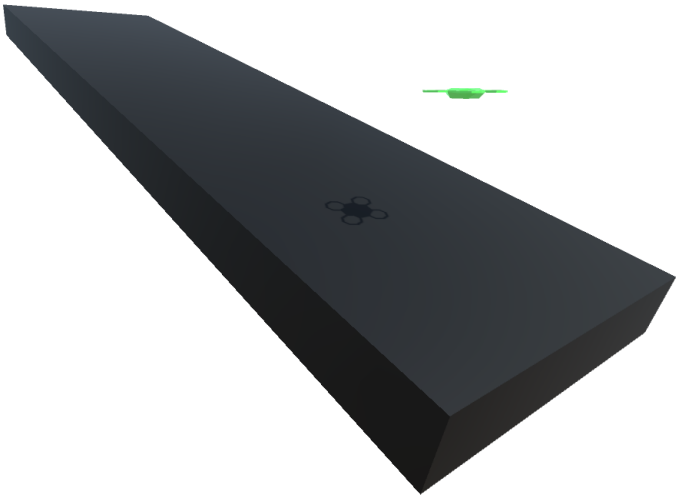


Figure 4.6. The environment of situation 3. The UAV approaches a slope, trying to move forward. The UAV has to either stop or move along the slope in order to be able to avoid collision. The UAV will also attempt to move down the slope.

wall given a wide enough horizontal FOV. Also note that there are invisible walls on the top and bottom of the wall, ensuring that the UAV cannot move over or under the wall.

The angles are 15°, 30° and 45°. The sensor windows tested are 110/110/15 and 58/45/3.4. Additionally, we test performance given the additional safety distance at 0m and 1m for each sensor window.

4.2.4 Situation 3: Incline/Downward Slope

This situation is divided into two parts. In the first part, the UAV is heading towards an incline of different vertical angles while attempting to move forward. It is expected to be able to move along the slope upward given a wide enough vertical FOV. In the second part, the UAV is heading towards a downward slope of different vertical angles while attempting to move down (and slightly forward). Also note that there are invisible walls on the sides of the incline/slope, ensuring that the UAV cannot move around it.

The angles are 15°, 30° and 45°. The sensor windows tested are 110/110/15 and 58/45/3.4. Additionally, we test performance given the additional safety distance at 0m and 1m for each sensor window.

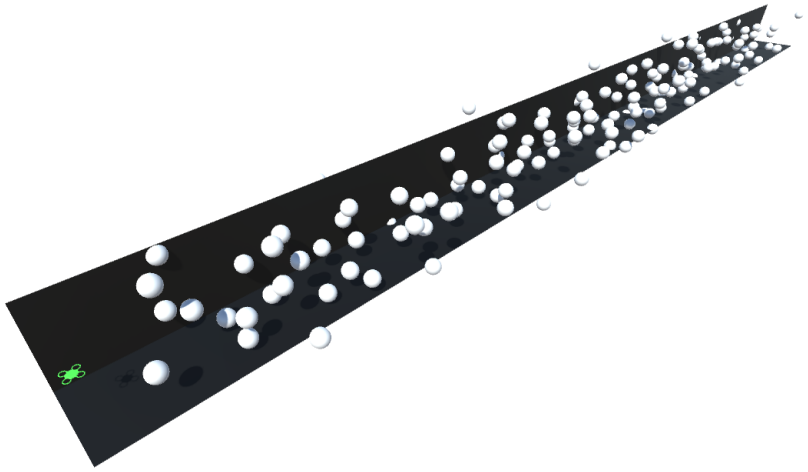


Figure 4.7. The environment of situation 4. The UAV attempts to move through a cluttered corridor.

4.2.5 Situation 4: Obstacle course

This situation is designed to be the most difficult for the UAV platform to traverse. The UAV is placed in front of a narrow corridor containing spherical obstacles sufficiently spaced between each other and the UAV is expected to be able to traverse the narrow corridor while avoiding the obstacles. This is not intended to reflect a realistic scenario, but instead to be able to evaluate the ability of the UAV to avoid obstacles in all three dimensions simultaneously.

Many different aspects regarding the obstacle avoidance method is tested. The tests are used to find out how different parameters affect the behaviour of the obstacle avoidance method. The parameters tested include the safety radius and additional safety distance, burn duration, update frequency, depth map resolution and sensor window. For each test that does not involve tests of the sensor window, the default sensor window is 360/180/15.

4.3 Default hyperparameters

There are a large number of hyperparameters that affect the performance and behaviour of the obstacle avoidance algorithm. Some hyperparameters related to the simulation of the UAV and sensors include mass, drag, time constants and maximum acceleration constraints. Some hyperparameters related to the obstacle avoidance algorithm is the safety radius, burn time, delta time resolution, etc. Other hyperparameters that may be affected due to computational constraints include the time

UAV motion model parameters	
Mass (m)	1kg
Drag coefficient (Constant D)	0.4
$\dot{v}_{\max}$	2m/s^2
$\tau_{x,y,z,\psi}$	1
Algorithm-specific parameters	
Desired velocity magnitude ($\|v_{\text{des}}\|$)	2m/s
Safety radius (s)	1m
Additional safety distance (d)	1m
Burn time (b)	0.4s
Delta time between states (dt)	0.1s
Angular width/height per pixel (α)	4°
Rudimentary control smoothing term (δ)	0.4
Sensor noise parameters	
Angular width/height between observations	2°
Sensor noise linear variance factor (λ)	10^{-6}
Lower uniform distribution bound (a)	0.2
Upper uniform distribution bound (b)	1
Update Frequency	10Hz

Table 4.1. Default hyperparameters for the tests.

between updating the depth map, depth map resolution, etc. Due to the number of different hyperparameters not all combinations can be tested exhaustively. Instead a default configuration of hyperparameters will be defined and apply for all tests unless an explicit change in parameters is defined. The default hyperparameters can be seen in Table 4.1

Chapter 5

Results

The results for situations 1 to 3 are presented by a pair of images. The first image presents a visualization of the trajectory of the UAV, where red UAVs represent the position and pose of the UAV over time, with one second between them. The second graph represents the distance to the obstacles over time.

For situation 4, the results are represented in Tables where the time to complete, average velocity, average distances lower than the safety distance and smallest recorded distance are presented for brevity.

5.1 Situation 0

There are clear differences between the sensor windows 360/180/15 and 58/45/3.4 when it comes to the ability of the UAV to move in a desired direction.

In the case of sensor window 360/180/15, the UAV was able to move forwards, backward, up, down and right while maintaining a desired velocity magnitude without issue. Trivially, the rudimentary control solution was evaluated as being permissible in these cases and there was no need for any modified control input.

The behaviour for sensor window 58/45/3.4 was more interesting. The UAV was able to move forwards, but it did not reach the desired velocity magnitude of 2m/s, instead reaching velocities closer to 1.5m/s. The obstacle avoidance method did not evaluate the rudimentary control solution as permissible given the sensory data, so other control inputs were tested. The UAV did not move when attempting to move backwards, which was expected given that there was no sensory coverage backwards.

The UAV was able to move up and down, however only at an angle. Since there was no sensory coverage up or down, the UAV could not evaluate the rudimentary control solution as permissible and instead had to evaluate a range of other control solutions. However, the angle at which the the UAV attempted to move may not have been desirable. The angle was not the smallest possible angle, but instead had a clear tendency of also moving sideways. A possible reason for this behaviour is that since the heuristic cost function values small angular errors and maintaining a

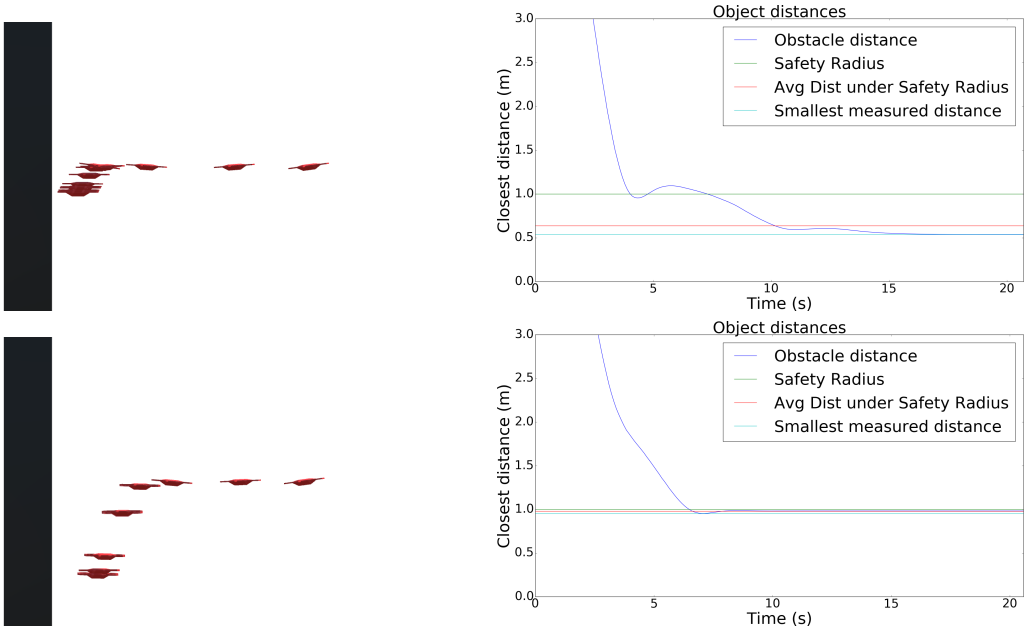


Figure 5.1. Situation 1, mid, using sensor window 110/110/15 with an additional safety distance of 0 (top) and 1 (bottom). The graphs show the closest distance to obstacles over time.

velocity equally, the control inputs which allowed the UAV to move sideways were used since it was able to allow the UAV to move at a velocity closer to the desired velocity magnitude. Additionally, when moving downward the UAV stopped at frequent intervals. Attempting to move right demonstrated similar issues.

The results of the trials suggest that if the sensor window is limited, and given the heuristic cost function, the UAV is able to manoeuvre even if there is no sensor coverage in the desirable velocity direction, although the positions that the UAV could move towards were constrained by the sensor window, which was expected. This may or may not be a desirable outcome, depending on whether it is acceptable to only move in a general direction, or the direction of velocity must be maintained. Only allowing for desirable velocities covered in the sensor window may result in more consistent (and predictable) movements.

5.2 Situation 1

5.2.1 Mid level

The results can be seen in Figures 5.1, 5.2. First, the UAV attempts to move towards the wall at mid level. With sensor window 110/110/15 and no additional safety distance, the UAV first stops close to the wall outside the safety radius, but then moves closer to the wall at an extreme angle, thereby moving to a point with a minimum obstacle distance less than the safety radius. With an additional safety

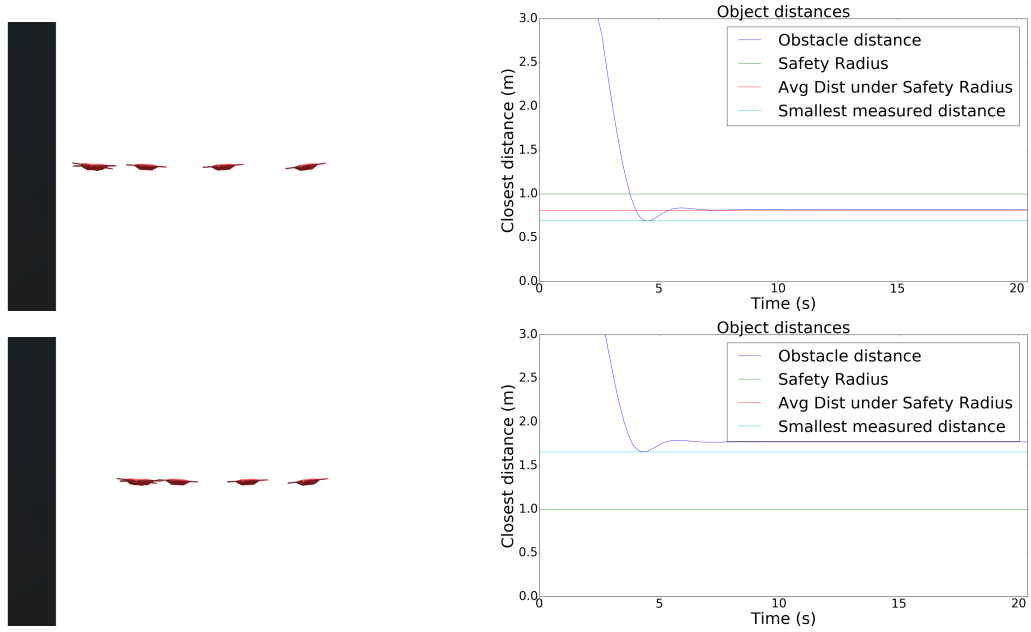


Figure 5.2. Situation 1, mid, using sensor window 58/45/3.4 with an additional safety distance of 0 (top) and 1 (bottom). The graphs show the closest distance to obstacles over time.

distance, the UAV again stops before the wall and moves closer to the wall at an extreme angle, this time being able to stop at a minimum obstacle distance close to the safety distance. A possible reason for this behaviour is discussed in section 6.2.3.

With sensor window 58/45/3.4, the UAV stops at a distance close to the safety radius without moving incrementally closer at an extreme angle. With an additional safety distance, the UAV is able to stop well before the wall, with a distance larger than the safety radius. A possible reason for this behaviour is discussed in 6.2.3.

5.2.2 Top level

The results can be seen in Figures 5.3, 5.4. Here, the UAV is closer to the level of the upmost edge of the wall, but not entirely. With sensor window 110/110/15, the UAV is able to find a path over the obstacle. However it is not able to maintain the distance to the obstacle without the additional safety distance.

For sensor window 58/45/3.4, the UAV is not able to find a way over the obstacle without the additional safety radius. With the additional safety radius, it is able to do so. One reason for this behaviour is that the additional safety radius forces the UAV to avoid obstacles earlier, which leads to the UAV moving up earlier. The ability of the UAV to move up decreases when approaching the wall, which the UAV is able to avoid.

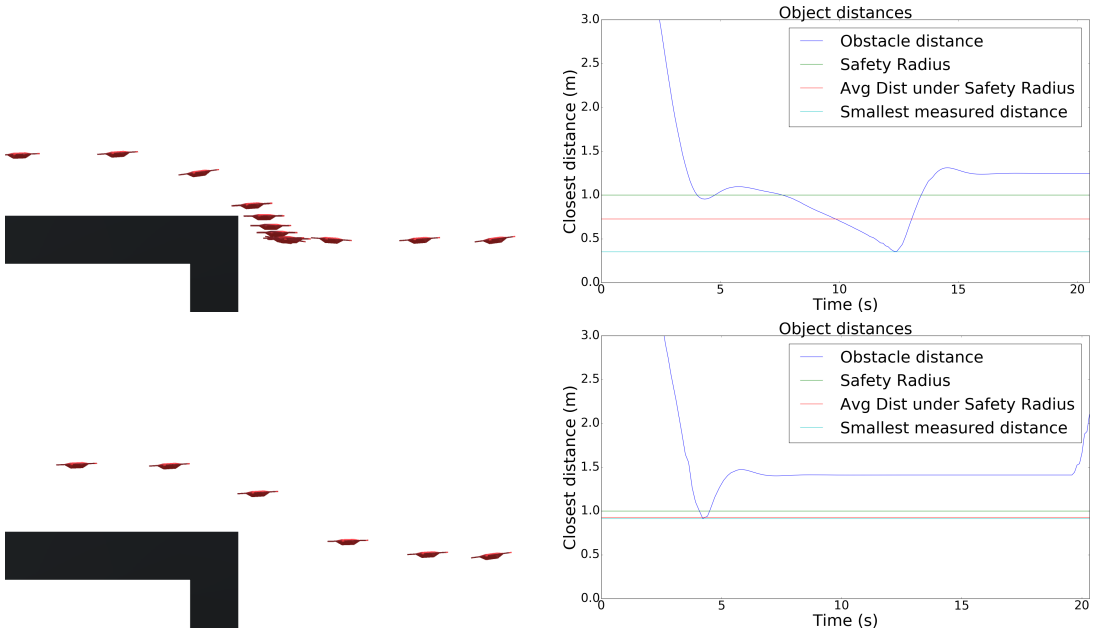


Figure 5.3. Situation 1, top, using sensor window 110/110/15 with an additional safety distance of 0 (top) and 1 (bottom). The graphs show the closest distance to obstacles over time.

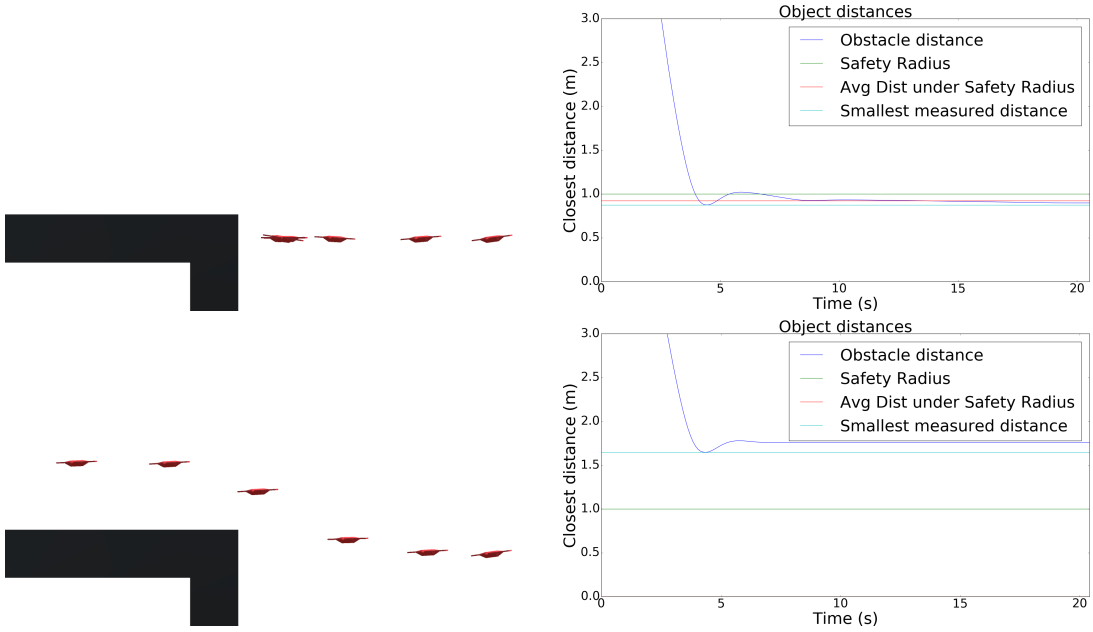


Figure 5.4. Situation 1, top, using sensor window 58/45/3.4 with an additional safety distance of 0 (top) and 1 (bottom). The graphs show the closest distance to obstacles over time.

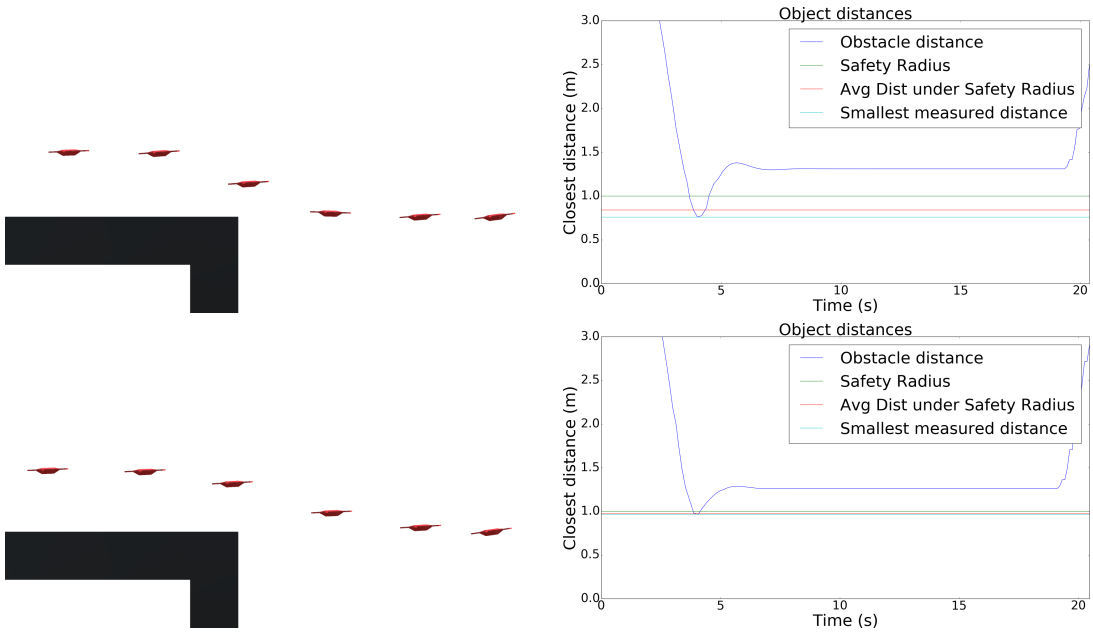


Figure 5.5. Situation 1, edge, using sensor window 110/110/15 with an additional safety distance of 0 (top) and 1 (bottom). The graphs show the closest distance to obstacles over time.

5.2.3 Edge level

The results can be seen in Figures 5.5, 5.6. Here, the UAV is on the same level as the upmost edge of the wall. Unsurprisingly, with sensor window 110/110/15, the UAV is able to avoid the obstacle. With sensor window 58/45/3.4, the UAV is able to navigate over the obstacle. However it is not able to maintain a distance to the obstacle, although the additional safety distance improves the result.

5.3 Situation 2

5.3.1 15 degree angle

The results can be seen in Figures 5.7, 5.8. With both sensor windows, the UAV was able to follow the wall and maintain a distance. There is no obvious difference between the results with and without the additional safety distance.

5.3.2 30 degree angle

The results can be seen in Figures 5.9, 5.10. Here some noticable difficulties following the wall can be seen. For sensor window 110/110/15, the UAV is able to follow the wall and maintain a distance reasonably well with and without the safety radius, but begins to drift downwards. For sensor window 58/45/3.4, without additional

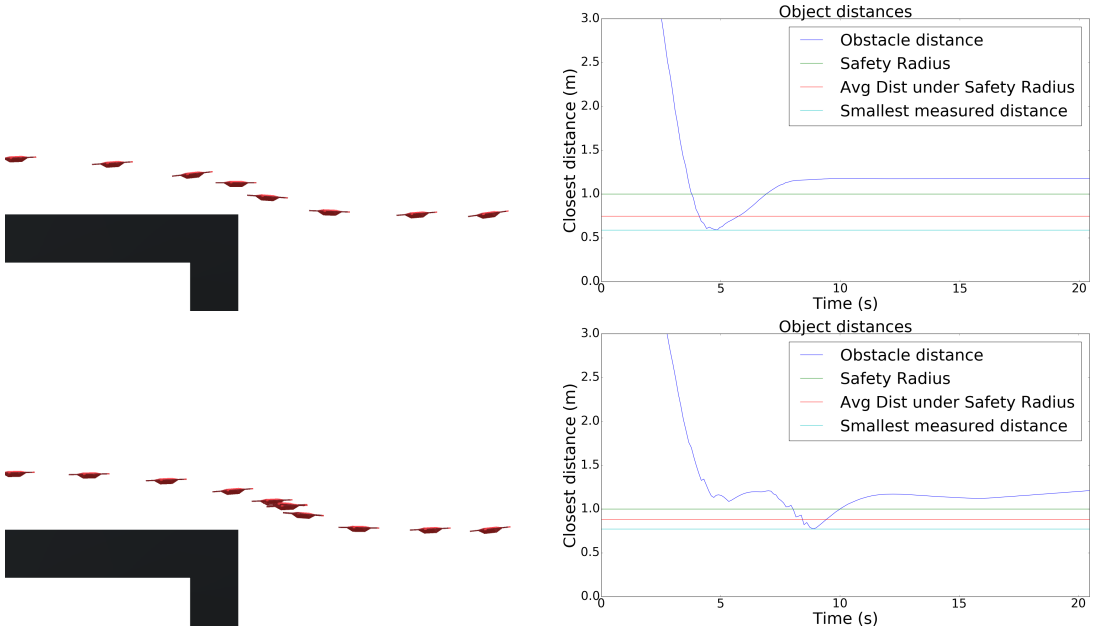


Figure 5.6. Situation 1, edge, using sensor window 58/45/3.4 with an additional safety distance of 0 (top) and 1 (bottom). The graphs show the closest distance to obstacles over time.

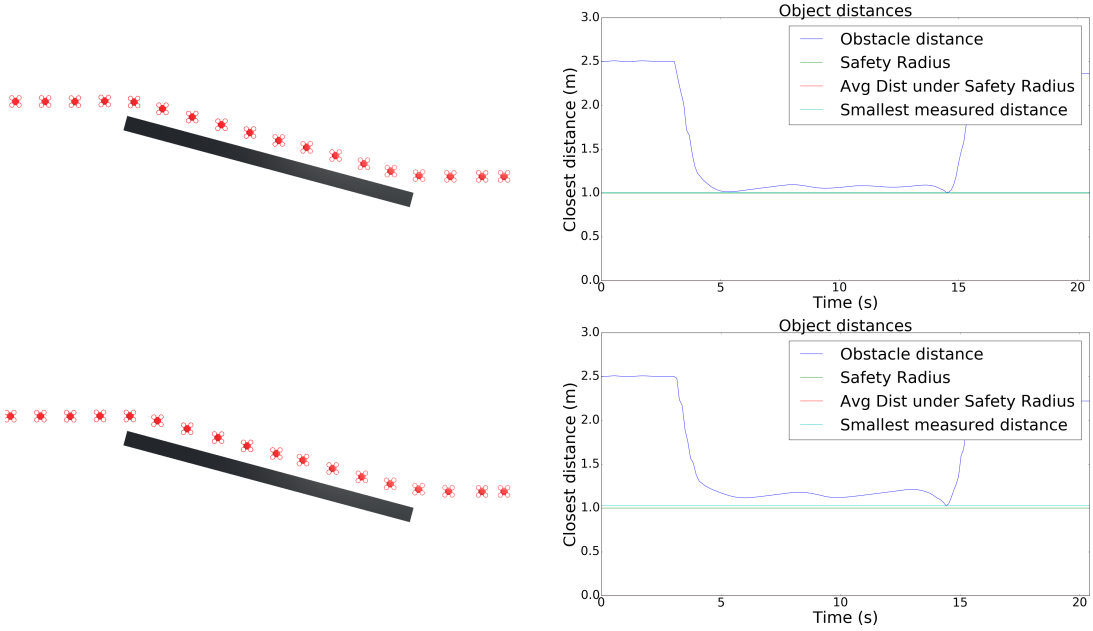


Figure 5.7. Situation 2, 15°, using sensor window 110/110/15. The graphs show the closest distance to obstacles over time.

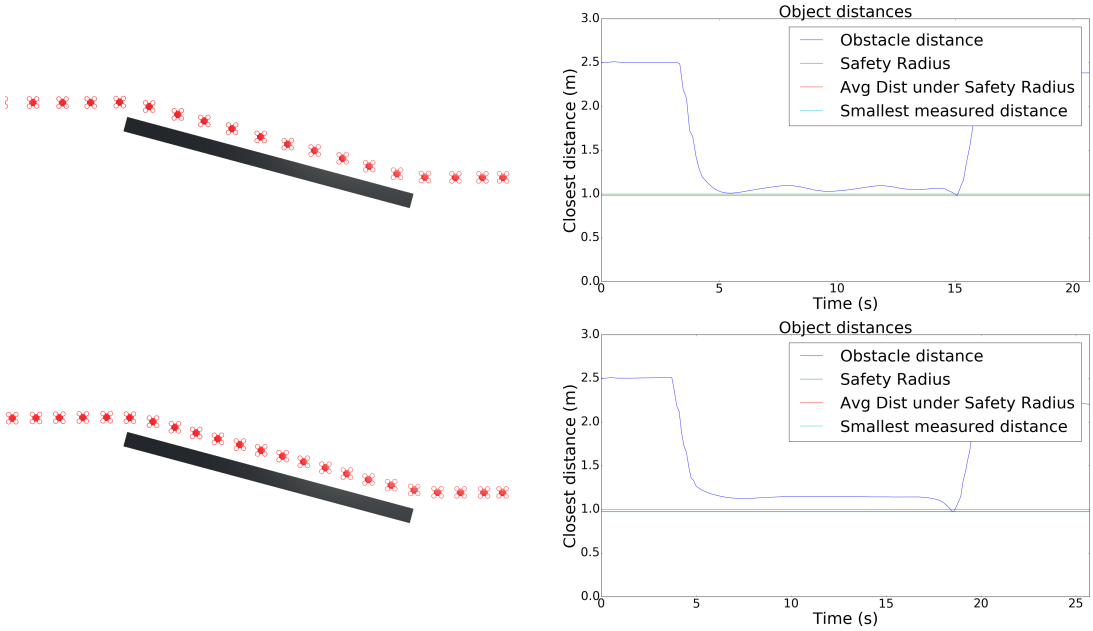


Figure 5.8. Situation 2, 15°, using sensor window 58/45/3.4. The graphs show the closest distance to obstacles over time.

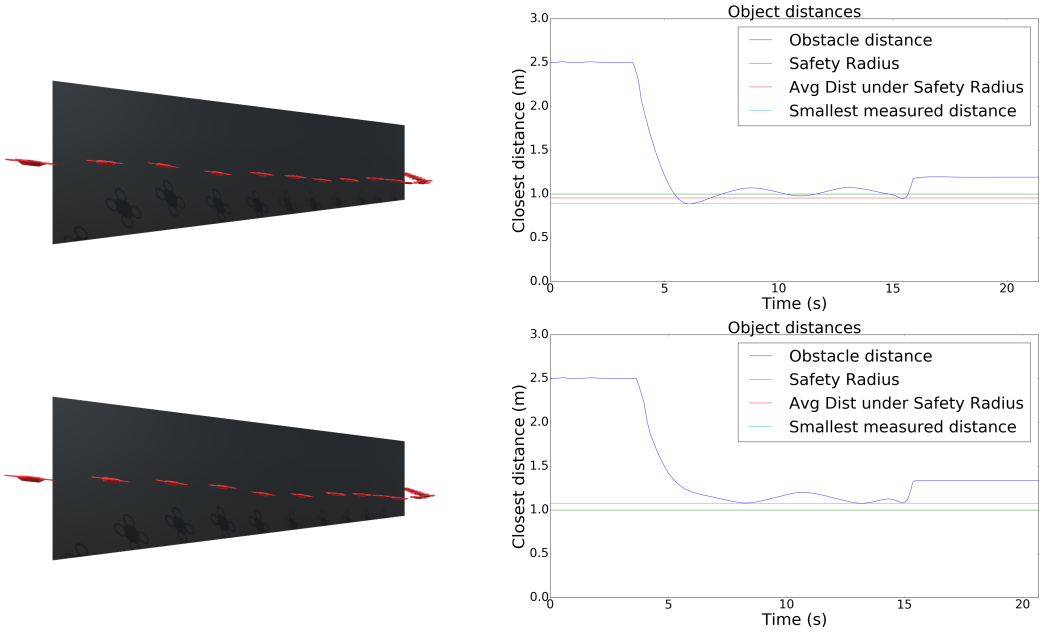


Figure 5.9. Situation 2, 30°, using sensor window 110/110/15. The graphs show the closest distance to obstacles over time.

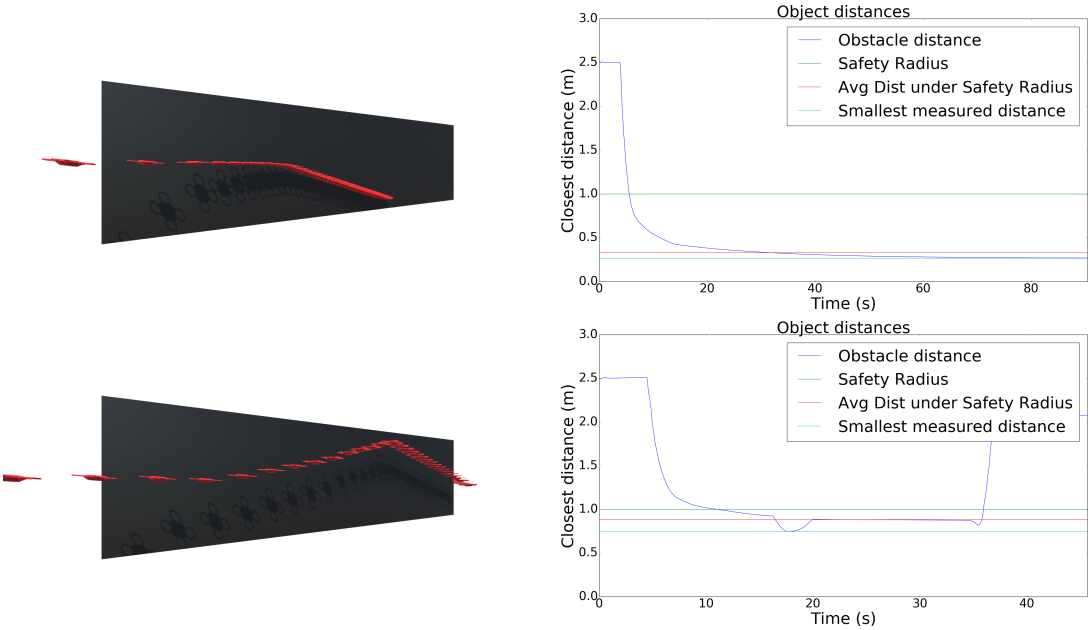


Figure 5.10. Situation 2, 30°, using sensor window 58/45/3.4. The graphs show the closest distance to obstacles over time.

safety distance, the UAV begins to drift downwards at a very slow velocity while approaching the wall, before finally stopping at an undesirable distance. With the additional safety distance, the UAV is drifting up and down, but is able to follow the wall closer to a desirable distance. The reason for the upward/downward drifting may depend on the heuristic cost function preferring directions which lead to faster velocities rather than smaller velocity vector angles. A possible reason for the UAVs tendency to approach the wall further is discussed in 6.2.3.

5.3.3 45 degree angle

The results can be seen in Figures 5.11, 5.12. For sensor window 110/110/15, without the additional safety distance, the UAV is able to follow the wall, although at a very slow velocity and not at a desirable distance. With the additional safety distance, the UAV is able to follow the wall at a desirable distance, although it drifts downwards.

For sensor window 58/45/3.4, the UAV is not able to follow the wall and stops. Without the additional safety distance, the UAV is not able to maintain a desirable distance to the wall, but is able to do so with the additional safety distance.

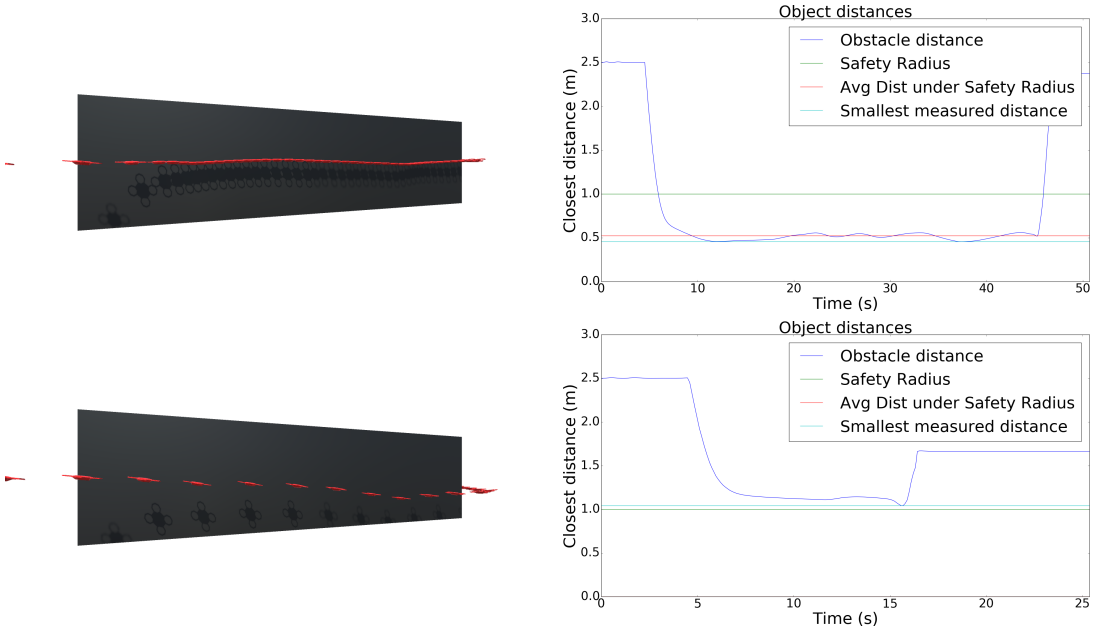


Figure 5.11. Situation 2, 45°, using sensor window 110/110/15. The graphs show the closest distance to obstacles over time.

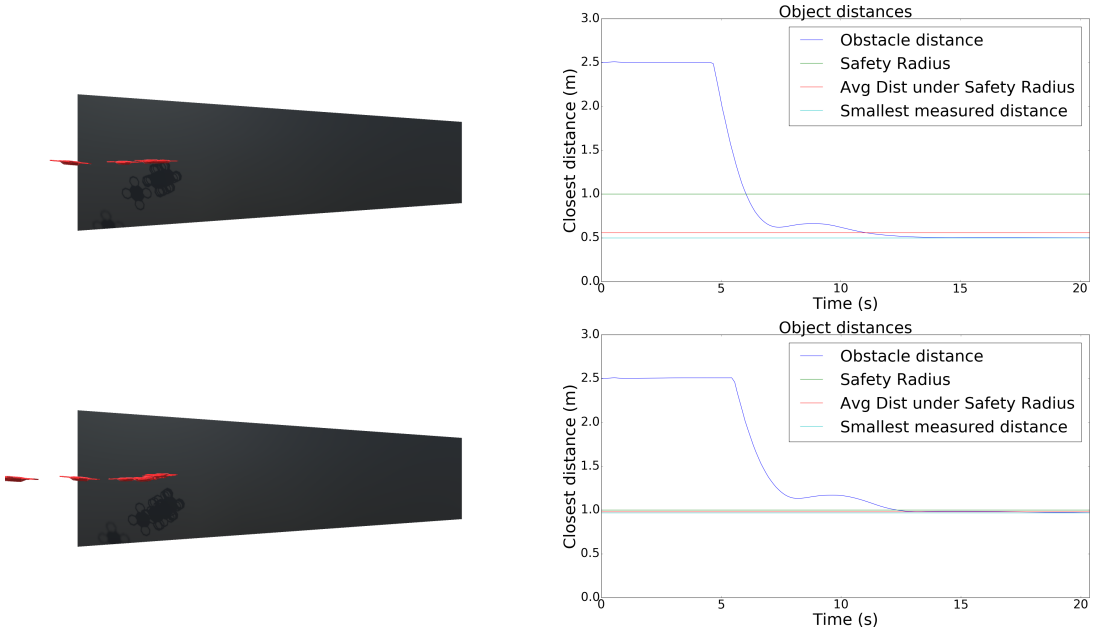


Figure 5.12. Situation 2, 45°, using sensor window 58/45/3.4. The graphs show the closest distance to obstacles over time.

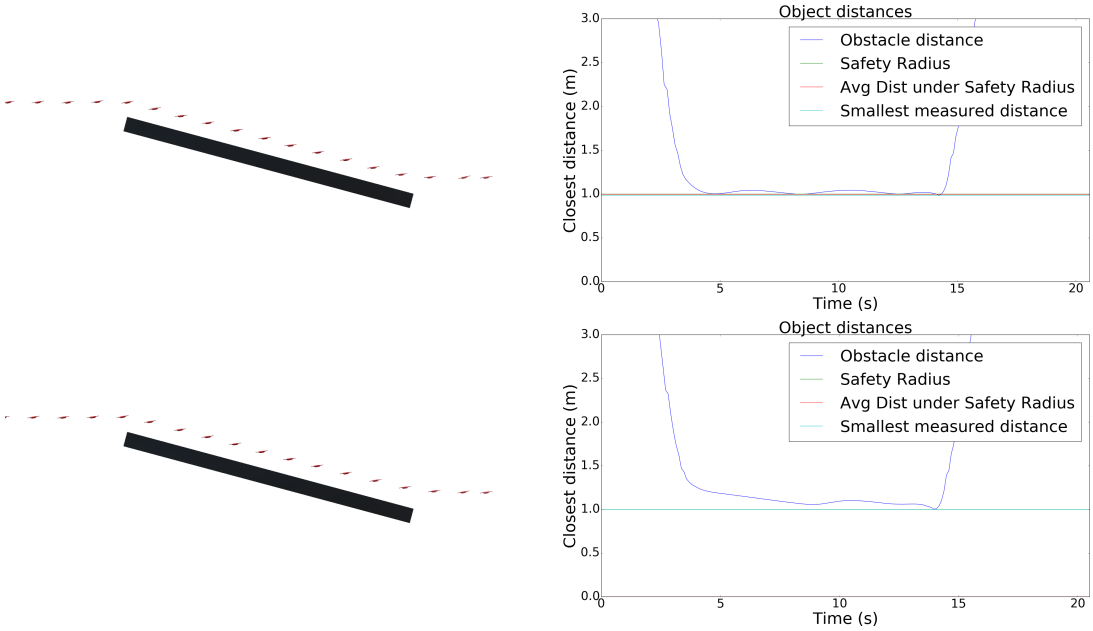


Figure 5.13. Situation 3, incline 15°, using sensor window 110/110/15. The graphs show the closest distance to obstacles over time.

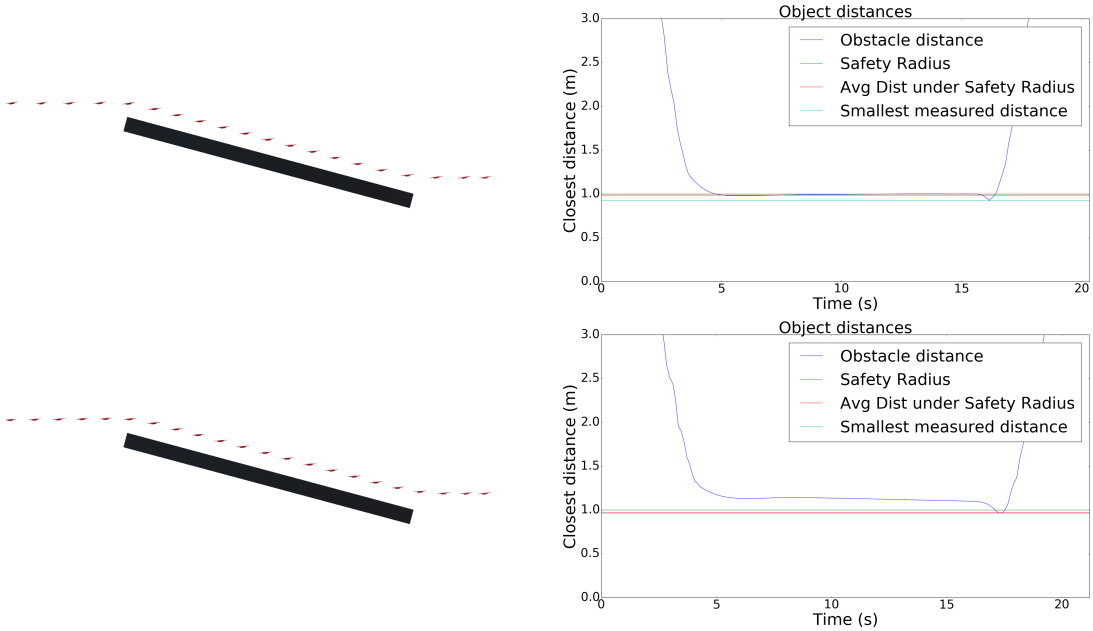


Figure 5.14. Situation 3, incline 15°, using sensor window 58/45/3.4. The graphs show the closest distance to obstacles over time.

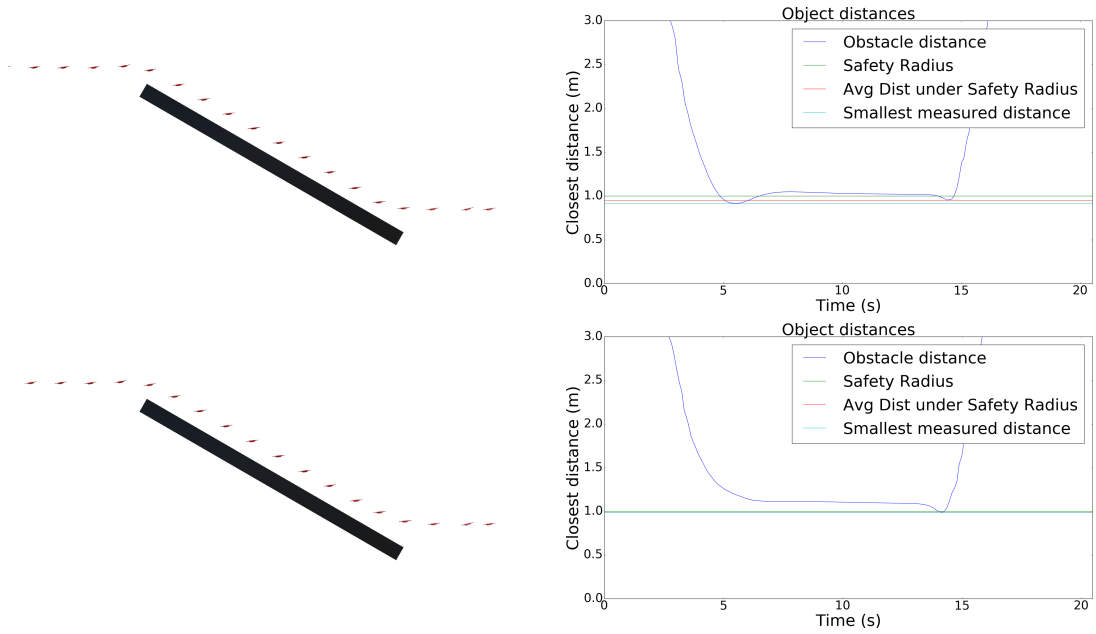


Figure 5.15. Situation 3, incline 30°, using sensor window 110/110/15. The graphs show the closest distance to obstacles over time.

5.4 Situation 3, Moving forward

5.4.1 15 degree angle

The results can be seen in Figures 5.13, 5.14. For both sensor windows, the UAV is able to follow the slope upwards at a desirable distance, with or without the additional safety distance.

5.4.2 30 degree angle

The results can be seen in Figures 5.15, 5.16. For sensor window 110/110/15 with and without the additional safety distance, the UAV is able to follow the slope upwards at a desirable distance. For sensor window 58/45/3.4, the UAV is unable to follow the slope upwards. Without the additional safety distance, the UAV is unable to maintain a distance to the slope, while with the additional safety distance, the UAV is able to maintain the distance.

5.4.3 45 degree angle

Since the UAV could not follow the slope upward with sensor window 58/45/3.4, it could not do so if the slope angle was at 45°, so this test was skipped for this sensor window. The result for sensor window 110/110/15 can be seen in Figure 5.17. The UAV is able to follow the slope upward without the additional safety distance, although at an undesirable distance to the obstacle. Interestingly, after

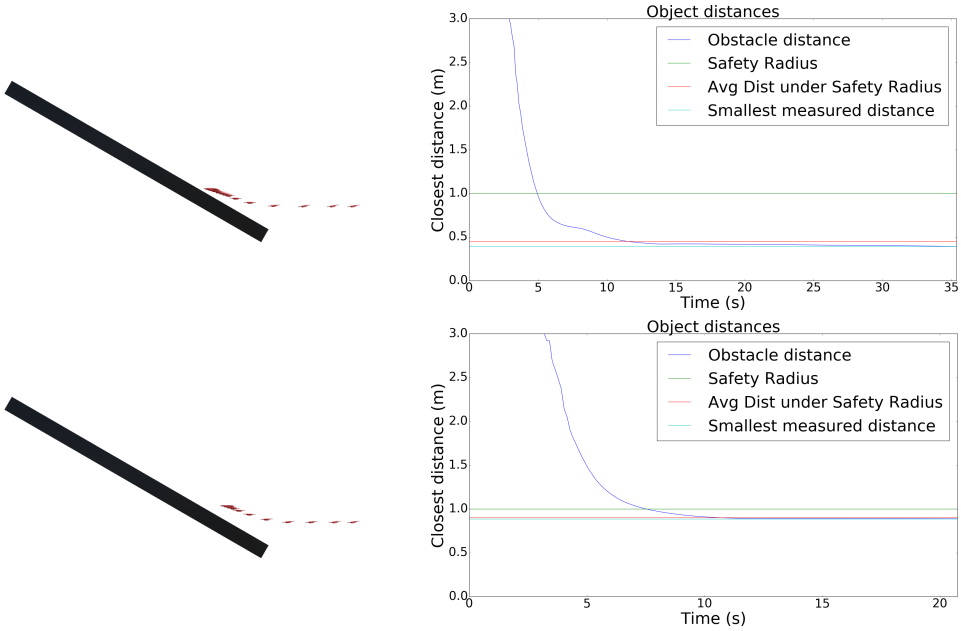


Figure 5.16. Situation 3, incline 30°, using sensor window 58/45/3.4. The graphs show the closest distance to obstacles over time.

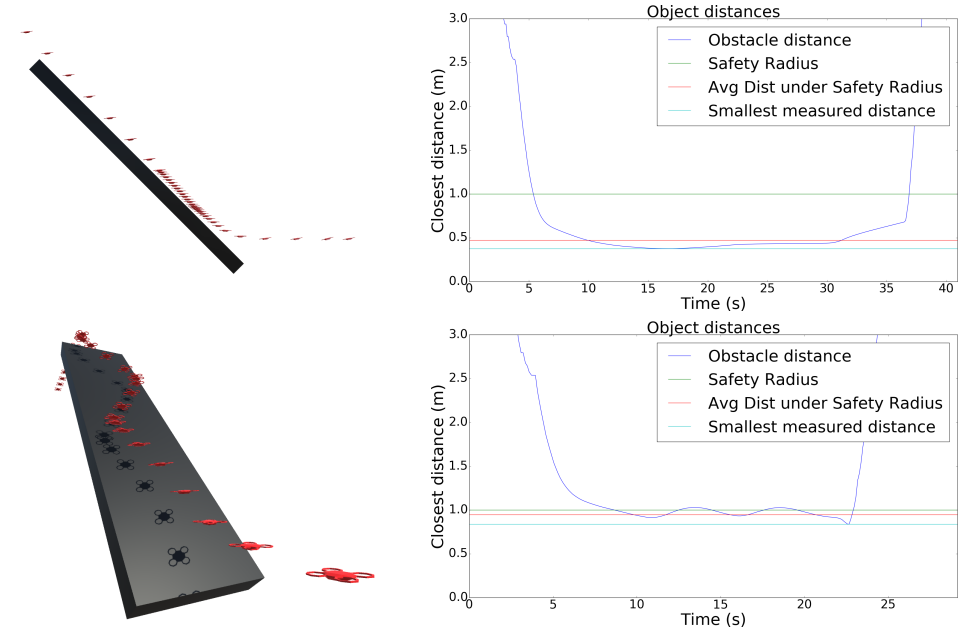


Figure 5.17. Situation 3, incline 45°, using sensor window 110/110/15. The graphs show the closest distance to obstacles over time.

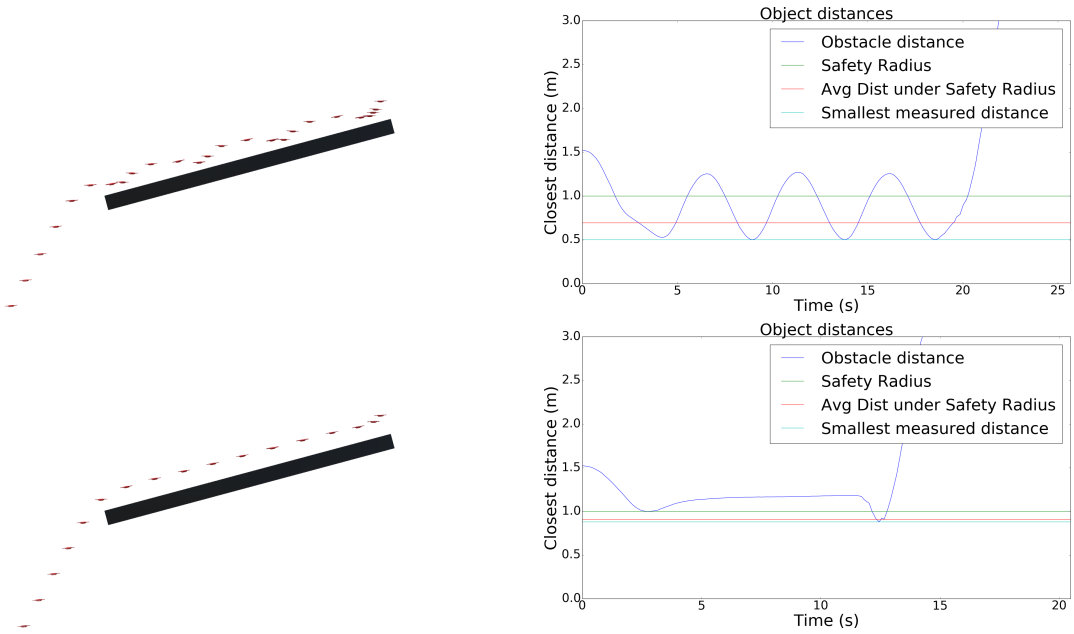


Figure 5.18. Situation 3, descent 15°, using sensor window 110/110/15. The graphs show the closest distance to obstacles over time.

the UAV has followed the incline, the UAV would be able to move in a more optimal manner in regards to the desired velocity. However, the UAV continues to maintain a suboptimal velocity. The reason for this behaviour is probably due to the heuristic function: for some reason, the rudimentary control solution, which would lead to a more optimal velocity, is found to not be permissible, and other control inputs are tested. In order to reach a more optimal velocity, some intermediate steps have to be taken which have a lower heuristic cost. Given the greedy nature of the command selection, these intermediate steps are never selected and as such the trajectory never improves. This situation could possibly have been avoided if the heuristic cost function did not penalize velocity magnitude errors as much.

This was not the case with the additional safety distance, however. With the additional safety distance, the UAV moved left and right instead while moving upward, also being able to maintain a distance to the slope. After avoiding the slope, the UAV was able to reach a more optimal velocity.

5.5 Situation 3, Moving down

5.5.1 15 degree angle

The results can be seen in Figures 5.18, 5.19. For sensor window 110/110/15, without the additional safety distance, the UAV adopted a "bouncing" motion, unable to maintain a stable distance to the obstacle. With the additional safety

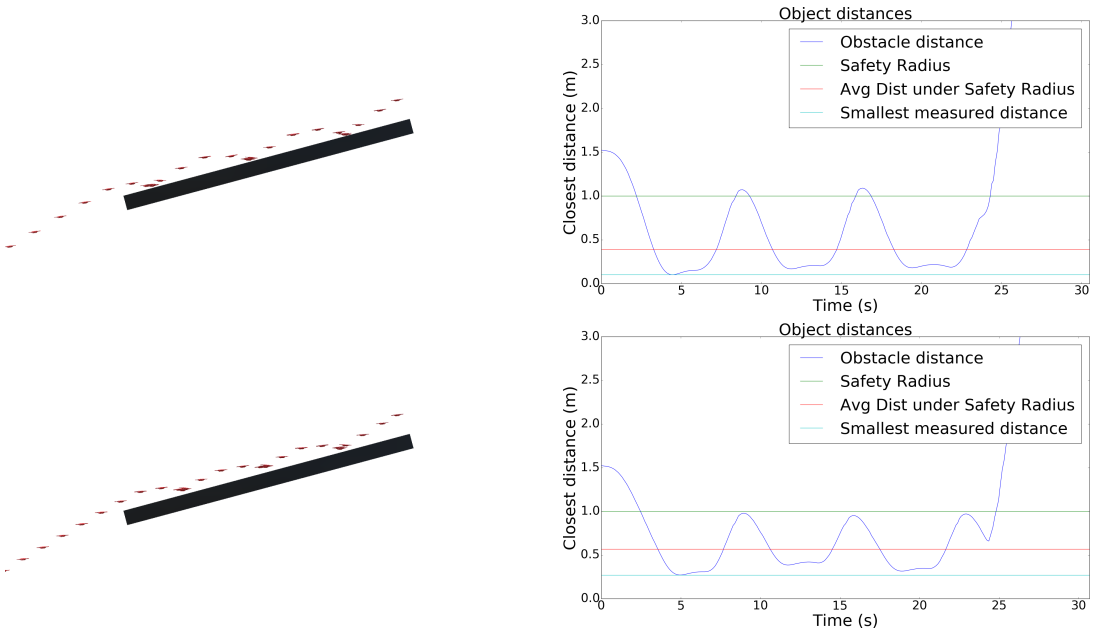


Figure 5.19. Situation 3, descent 15°, using sensor window 58/45/3.4. The graphs show the closest distance to obstacles over time.

distance, the UAV was able to maintain a stable, desirable distance to the obstacle. This was not the case with sensor window 58/45/3.4, where a stable distance could not be maintained. The additional safety radius only improved the results modestly.

5.5.2 30 degree angle

The results can be seen in Figures 5.20, 5.21. For sensor window 110/110/15, the UAV could not maintain a stable distance to the obstacle, with or without the additional safety distance. For sensor window 58/45/3.4, the UAV maintains its distance to the obstacle more consistently. However, this is most likely due to the limited FOV, where the theoretical maximum possible traversable slope at resting position is 22.5°, only increased slightly by the forward movement of the UAV. The UAV is therefore unable to actually move close to the obstacle.

5.5.3 45 degree angle

With sensor window 58/45/3.4, the UAV is unable to approach the obstacle and as such, the result is uninteresting and therefore omitted.

The results with sensor window 110/110/15 can be seen in Figures 5.22. With sensor window 110/110/15, the UAV is able to follow the wall but is still unable to maintain an acceptable distance to the obstacle. The additional safety distance only has slightly better results. Furthermore, the UAV continues with a suboptimal

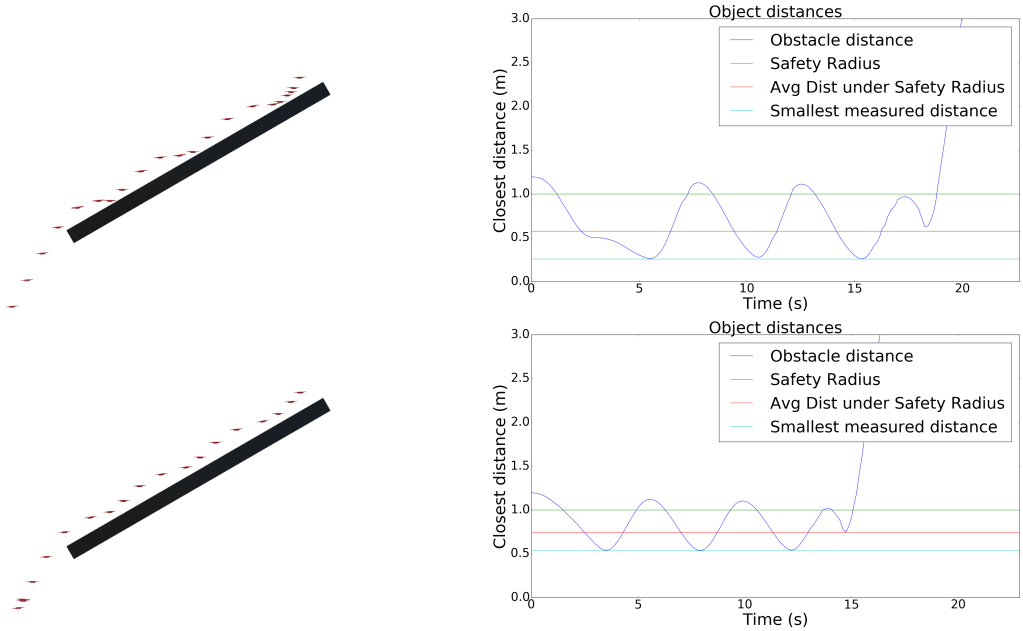


Figure 5.20. Situation 3, descent 30°, using sensor window 110/110/15. The graphs show the closest distance to obstacles over time.

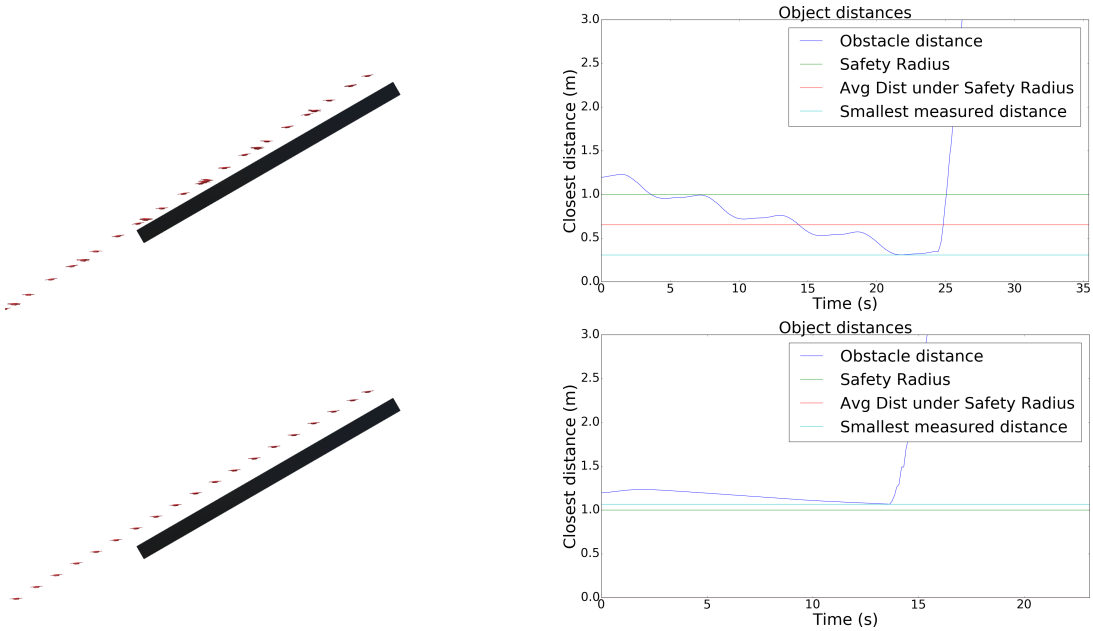


Figure 5.21. Situation 3, descent 30°, using sensor window 58/45/3.4. The graphs show the closest distance to obstacles over time.

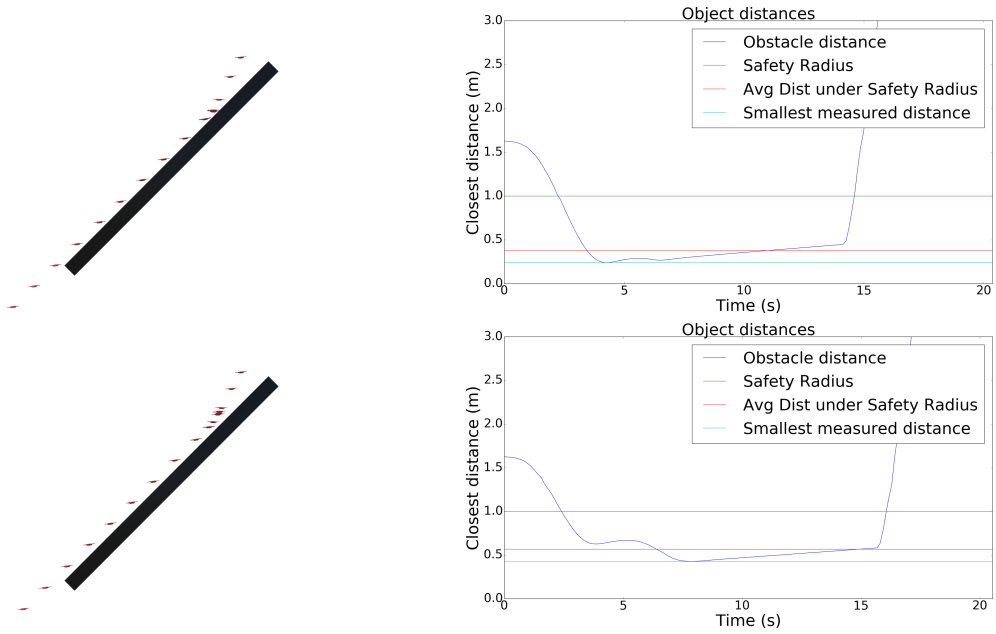


Figure 5.22. Situation 3, descent 45°, using sensor window 110/110/15. The graphs show the closest distance to obstacles over time.

s / d (m)	Time to complete (s)	Average velocity (v)	Avg. dist. under safety r. (m)	Minimum distance (m)
0/0	51.4	1.95	0.56	0.01
0/1	52.8	1.88	0.58	0.0
1/0	62.7	1.63	0.84	0.63
1/1	67.9	0.46	0.91	0.79

Table 5.1. Results for different combinations of safety radii and additional safety distances.

velocity after the obstacle has been passed. Again, this is believed to be caused by a local minima in the heuristic cost function.

5.6 Situation 4

5.6.1 Safety radius and additional safety distances

The results can be seen in Table 5.2. Given that the safety radius is 0 and no additional safety distance, the UAV is essentially treated as a singular point and no post-processing step is taken. The apparent behaviour with no safety radius is that the UAV does evade frontal collisions, but instead moves towards the edges of the spherical obstacles. The end result is many close encounters where the minimum

Sensor window	Time to complete (s)	Average velocity (v)	Avg. dist. under safety r. (m)	Minimum distance (m)
360/180/15	67.9	1.52	0.91	0.76
110/110/15	65.2	1.57	0.93	0.83
58/45/3.4	82.5	1.24	0.87	0.49

Table 5.2. Results for different sensor window

distance to obstacles is close to zero. The inclusion of an additional safety distance does little to change this fact, since this treats the dimensions of the UAV as a one-dimensional line and does not avoid obstacle observation to its sides.

The safety radius and post-processing step is absolutely crucial in order to avoid obstacles. The UAV is able to consistently maintain a distance to obstacles, although not necessarily outside the safety radius. The results can be improved further with the additional safety distance.

Due to the increased performance with the additional safety distance, the following runs utilize an additional safety distance of 1m.

5.6.2 Sensor windows

The results can be seen in Table 5.2. All sensor windows managed to complete the obstacle course and maintain a decent distance to obstacles. The difference between the omnidirectional sensor window and 110/110/15 is not significant, due to the omnidirectionality not being necessary when only moving in one general direction.

There were some notable differences between the larger sensor windows and sensor window 58/45/3.4. For this sensor window, the time to complete was higher, but this could be explained by the lower maximum velocity. However, unlike for the other sensor windows, which were able to maintain a velocity during the entire run, the UAV stopped on a number of occasions before being able to move. The ability to maintain a distance to obstacles was also slightly lower and the UAV had one unfortunate close encounter with an obstacle.

5.6.3 Burn Time

Results can be found in Table 5.3. Lower burn time is associated with faster reactions and acceleration changes but also later reaction times. The result is that the UAV moves towards obstacles at relatively high velocities before braking violently. The result is overall faster times to complete the obstacle course, but may also result in more frequent close encounters. The data also suggests that higher burn times result in a more consistent obstacle avoidance behaviour. However, this seems to reach its peak at 0.8s and subsequent increase seems to yield worse results, most significantly in time to complete. The difference in time to complete between 0.4s

b (s)	Time to complete (s)	Average velocity (v)	Avg. dist. under safety r. (m)	Minimum distance (m)
0.1	66.3	1.54	0.89	0.68
0.2	67.2	1.51	0.90	0.77
0.4	66.9	1.53	0.91	0.79
0.8	70.3	1.46	0.93	0.80
1.2	96.9	1.07	0.89	0.47
1.6	106.6	0.98	0.91	0.78

Table 5.3. Results for different burn times

Update freq. (Hz)	Time to complete (s)	Average velocity (v)	Avg. dist. under safety r. (m)	Minimum distance (m)
inf	64.4	1.58	0.92	0.80
10	67.9	1.49	0.91	0.77
2	73.1	1.42	0.93	0.78
1	83.7	1.23	0.92	0.66
$\frac{2}{3}$	104.7	1.0	0.89	0.70
0.5	121.4	0.89	0.88	0.44

Table 5.4. Results for different update frequencies

and 0.8s is much smaller than the difference in 0.8s and 1.2s. The reason for this behaviour is most likely due to small free space volumes when the UAV is close to obstacles. If the UAV is expected to maintain a velocity in a constrained space for a prolonged duration of time, it is increasingly unlikely for faster velocities to be permissible.

5.6.4 Update frequency

Results can be found in Table 5.4. Reductions in update frequency presents a clear trend in the time to complete the obstacle course. With higher update frequencies the UAV is able to complete the course faster. With an update frequency of at least 2Hz the UAV was able to complete the course without occasionally stopping. One reason for this is probably due to underestimation, which is discussed in section 6.2.3.

There was little difference in the ability of the UAV to maintain distances to obstacles. The update frequency does not necessarily affect the quality of the observations or ability to avoid the obstacles, although close encounters are more frequent for lower update frequencies.

α	Time to complete (s)	Average velocity (v)	Avg. dist. under safety r. (m)	Minimum distance (m)
2	61.5	1.65	0.89	0.75
4	69.8	1.48	0.92	0.78
6	75.4	1.36	0.92	0.77
8	80.5	1.28	0.90	0.76
12	107.4	0.97	0.90	0.68
18	152.9	0.68	0.80	0.53
24	218.7	0.50	0.77	0.15

Table 5.5. Results for different depth map resolutions

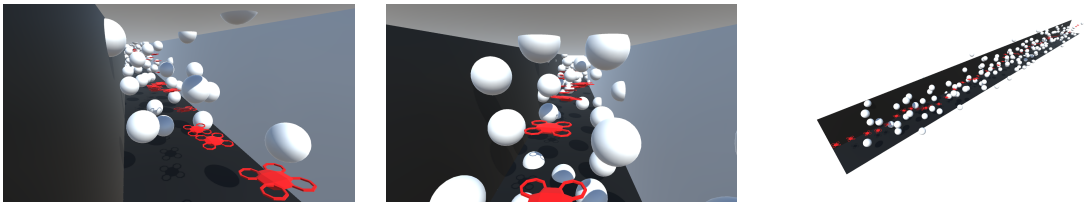


Figure 5.23. An example run of the UAV in the cluttered environment.

5.6.5 Depth Map resolution

Results can be found in Table 5.5. Due to the greater computational complexities incurred with higher resolutions, the sensor window that was used for these tests was 180/180/15. The most obvious result of lowering the resolution of the depth map is increased times to complete. The reason for this behaviour might be because a lower resolution will result in a smaller free space volume. Since each pixel in the depth map represents the smallest distance to an obstacle in a direction, each pixel will represent a larger area of angles the lower the resolution. When assigning values to the depth map, the closest distance in the area of directions to the UAV will be assigned each pixel. This will lead to a general underestimation of distances to obstacles, leading to a smaller free space volume.

Since the UAV underestimates distances to obstacles with lower resolutions, it should theoretically lead to the UAV generally maintaining greater distances to obstacles. This is obviously not the case for resolutions 18° - 24°, where overall obstacle avoidance performance goes down. Reasons for this behaviour may be related to the coarseness of the depth map and post-processing steps. Recall equation 3.6, which defines the boundary of the enlargement area around a pixel as $2\beta \times 2\beta$. If $\beta < \alpha$, then no enlargement of the neighbouring pixels is performed. The probability of this occurring increases as the resolution decreases (as α increases), which limits the ability to take the dimensions of the UAV into consideration. Overall, a too low resolution is not preferable.

Chapter 6

Discussion

In this chapter, we discuss the findings of this thesis, how it relates to the field of obstacle avoidance. Furthermore, we discuss some weaknesses of this approach, but also possible ways of rectifying them.

6.1 Comparisons with other methods

In this section, we compare the proposed obstacle avoidance method with other methods.

6.1.1 Optical flow methods

The obstacle avoidance methods based on optical flow typically only allow for two-dimensional obstacle avoidance due to the heuristic of balancing the flow between the left and right side of the UAV. The assumptions made on the UAV is that it is only able to move forward relative to its heading, leading to a loss of one degree of freedom. In order to change the direction of the linear velocity, the UAV has to alter its yaw, thereby altering the heading of the UAV. In essence, the UAV is assumed to have a differential drive motion model with the added ability to move in the lateral plane. In contrast, the proposed method in this thesis is able to avoid obstacles in all three dimensions.

The heuristic of balancing flow between the two sides may pose a challenge for manoeuvring close to obstacles. For example, if the UAV is expected to follow a wall with this principle, it will turn away from the wall and eventually fly away, even if the UAV flies parallel to the wall. As demonstrated, the proposed method is able to move along a wall while maintaining a set distance.

However, the proposed method in this thesis and optical flow are not mutually exclusive since it is possible to make distance estimates using optical flow, which is a fundamental requirement for the method. It may require more accurate flow models however as flow may be produced from a wide variety of motions.

6.1.2 Potential fields methods

Potential field methods could have been implemented given the depth map H. Since H represents the distances to obstacles in a given direction, H could be converted to a force vector map for some heuristic force function.

However, potential field methods fell out of favour as viable methods for avoiding obstacles. One issue with potential field methods is the demonstrated proneness to induce oscillatory motions when moving through narrow corridors as well as the occasional inability to move into narrow passages. Potential field methods also do not explicitly take into account the motion model of the platform which makes the overall behaviour difficult to predict, although multirotor UAVs are more suitable for such implementations due to their holonomic qualities in regards to translation.

In the context of occluded/unknown space and sensor windows, potential field methods may also behave completely contrary to the assumptions made. If the sensor window is not omnidirectional, then there is a good chance that there is no possible way to assess a potential counterforce for some direction. For example, if the UAV is close to a wall facing it, there may be an induced force which pushes the UAV away from the wall. However, if the sensor window has no coverage behind the UAV, then the UAV is pushed into unknown space, where occupancy cannot be assessed. It may therefore be pushed into an unseen obstacle. Potential field methods would therefore only be viable for omnidirectional sensor windows, or for directions where such counterforces could be assessed. However, potential field methods always consider obstacles in all observable directions which the proposed method does not. If the UAV enters occluded space (assuming omnidirectional sensory coverage) it is probable that the UAV would be passively pushed out of it, while the proposed method is only able to actively move out of occluded space and even then such attempts to move out of it may result in moving into occluded space further, as is discussed in 6.2.3. Potential field methods may therefore have an edge over the proposed method in certain circumstances.

6.2 Potential drawbacks

In this section, potential drawbacks of the proposed method is discussed.

6.2.1 Motion model complexity

The method presented requires knowledge of the motion model in order to be able to predict the motion of the UAV. The motion model used in this thesis is highly idealized and does not take into account actual motor dynamics and advanced aerodynamical effects. As such, other, more complex motion models may be required in a real setting. The definition on how to find the next state is intentionally general, so migrating the method to another motion model only requires redefining X, U and $f(X, U)$ to fit the motion model. If relevant, the rudimentary control solution also has to be redefined.

However, increased model complexity may carry computational implications. The method may scale poorly with increased control parameters since the method of finding viable control parameters is done by using brute force. Other methods such as some variant of RRT, or a focused search around the rudimentary control solution can be done instead.

However, increasing the model complexity is only justified if it is necessary. Some aerodynamical effects may not be relevant in some cases, such as if the velocity of the platform is low, or the force imposed on the platform can be changed rapidly. It could equally be the case that a decreased complexity is justified. In the end, the requirements on the motion model complexity depend on the ability of the method to accurately predict the path of the platform.

6.2.2 Mechanical limitations

There are a number of mechanical limitations that may be important to consider. These issues stem mainly from the relationship between the mechanics of the UAV motion model and the sensory coverage. One fundamental assumption made for the sensors was that they were rigid in relation to the UAV body, meaning that the orientation of the sensor window was determined by the orientation of the UAV.

In reality, omnidirectional sensor coverage is a utopia. Currently, there are no single sensors which can provide a complete omnidirectional FOV. Disregarding omnidirectional range-finders, there are a number of three-dimensional range-finders with a limited FOV. Since the sensor window dictates the volume which the UAV is allowed to move within, the FOV clearly constrain the mobility of the UAV. However, the act of moving horizontally also alters the pitch and roll of the UAV and therefore the rigid sensor. If a UAV moves diagonally forwards and up, the horizontal motion prompts the UAV to tilt forward, which reduces the lateral sensory coverage over the horizon, which results in an inability to move upward. Placing the sensors on a gimbal only solves one part of the problem, namely that the horizontal and lateral sensory coverage does not dramatically change depending on the horizontal movement of the UAV. The limited FOV still constrains the mobility of the UAV.

More pressing is the fact that obstacles outside the periphery of the sensor window still pose a risk for collision with the UAV. For low FOV range-finders, obstacle detection to the sides is limited and obstacles may evade detection. The problem of unseen obstacles could be one explanation for why the performance on the downward slope in situation 3 was poor. A possible heuristic solution is to always assume that obstacles are present just by the periphery. This will result in a lower free space volume, leading to even less room to manoeuvre.

6.2.3 Environment representation weaknesses

The proposed method of representing the environment as a two-dimensional depth image for obstacle avoidance purposes is reasonable. However, there are some issues

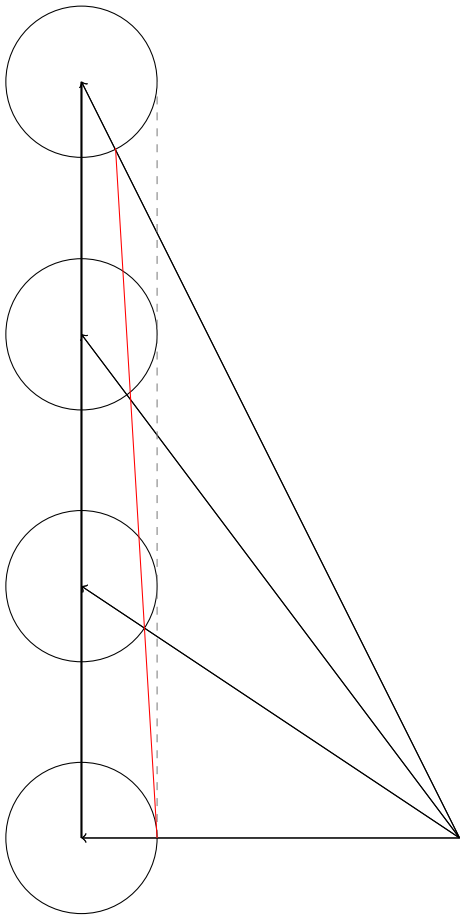


Figure 6.1. Ignoring the normal of the surface leads to overestimation of distances to obstacles. The gray dashed line represents the ideal margin while the red line is the approximate actual margin.

of the presented method which needs careful attention.

The method relies on the ability to assess the distance to obstacles in a given direction. The environment representation from the depth map H, assuming accurate measurements, is able to do so. However, by only considering the distances in a specific direction, the method would ignore obstacles which are not strictly positioned in the evaluated direction, but could nonetheless be relevant in order to accurately avoid obstacles. An example of a situation where this problem becomes apparent can be seen in Figure 6.1. Only considering the direction to the obstacles ignores the fact that we must also be able to avoid the obstacle in the direction of the surface normal. This results in an overestimation problem, where occluded space is erroneously classified as free, which may result in the platform moving closer to obstacles.

The post-processing step attempts to resolve this problem by assuming that there is a spherical boundary around all observations. As a consequence the actual

surface normal is irrelevant, as all possible normals are handled, which removes the need for normal estimation. However, the method by which this boundary is estimated may not have been accurate, and may therefore lead to some overestimation problems. Also, overestimation problems may be more prominent in the periphery of the sensory coverage, as there may be obstacles there which cannot be considered. In the situation where the platform moved towards a wall we observed that the platform would stop at a suitable distance but would then move further inward at an extreme angle. Similarly, in the downward slope scenario, which could easily be regarded as the situation where the method performed the worst, the platform attempted to move close along the periphery, and as a result a large part of the obstacle close to the UAV could not be seen. It may therefore be advisable to disallow motion in direction close to the periphery.

The overestimation error is very difficult to assess since it is often a result of missing information. In any case, the additional safety distance can be interpreted as a means to correct this error, if we assume that the overestimation error is constant. This may be the reason why the additional safety distance increased the reliability of the platform to maintain distances. However, including the additional safety distance does not remove strange movements based on this error.

The proposed method also underestimates some distances due to the post-processing step. An example can be seen in Figure 6.2. The reason is that the environment representation cannot discriminate between the spherical boundaries and the actual environment. In the image, it would be much more sensible if the boundary continued parallel to the direction to the observation, but the boundary will continue in the tangent direction instead. This results in false positives regarding the assessment if a position is occluded. This is not typically an issue if the update frequency is high, as the representation is accurate close to the UAV and the obstacle avoidance method is designed for local decisions. This, however, may lead to issues when the update frequency is very low, so a high update frequency is preferable. Alternatively, the UAV may operate at lower velocities instead, where the need for a high update frequency is lessened.

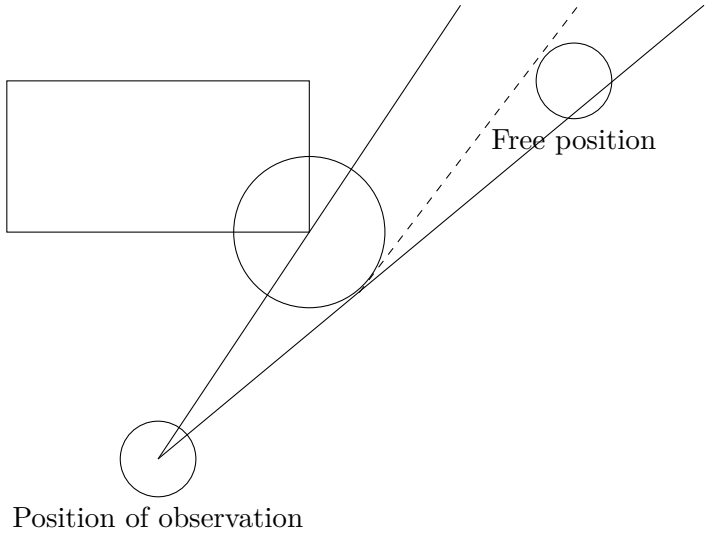


Figure 6.2. The post-processing step results in an underestimation of distances to obstacles. The dashed line portrays a more accurate margin where the free position can be classified as such. However, the information left after the enlargement post-processing step underestimates the possible distances to seen (or unseen) obstacles.

Chapter 7

Conclusion

This thesis has been dedicated to add to the understanding of obstacle avoidance in three-dimensional environments, mainly regarding obstacle avoidance for copter-based UAV platforms. In order to do so, an obstacle avoidance method was developed and tested. By representing the local environment as a depth map with each element representing a spherical coordinate, it was possible to evaluate if a position was either free or occluded. However, the original environment representation is not powerful enough for obstacle avoidance purposes. The representation overestimates the volume of free space when the dimensions of the UAV is considered. To solve this problem a post-processing step was introduced which allowed the UAV to better maintain distances to obstacles. However, the introduction of the post-processing step results in an underestimation error, though for obstacle avoidance purposes this error is benign. Also, the method may perform poorly when dealing with obstacles where the observations have a large angle of incidence relative to the surface normal.

Given that it is possible to predict the movement of the UAV for some control input, it is possible to evaluate if a control input will lead to the UAV entering occluded space. The method allows the multirotor to better maintain a distance to obstacles and is therefore a reasonable approach for obstacle avoidance purposes.

The heuristic cost function, which was used to select command inputs based on the ability of the command input to maintain a desirable direction and velocity magnitude equally, was found lacking in certain respects. By preferring these metrics equally, some strange behaviours were noticed that, while not greatly affecting overall obstacle avoidance performance, may prove disruptive and unpredictable to some.

Beyond the issues related to the obstacle avoidance method, there may be potential issues for rigid rangefinders as there is little room to control the orientation of the multirotor. At most the yaw can be altered without affecting the motion of the multirotor. This fact coupled with the implausibility of omnidirectional sensors results in limitations of possible manoeuvres.

7.1 Future work and extensions

The obstacle avoidance method was tested and evaluated in a simulated setting. While it allows for great control over every aspect of the test parameters, it does not necessarily correspond well for live applications. The UAV motion model, sensor model and test environment are highly idealized. Future work on this method should involve evaluating this method on real multirotor platforms, using real sensor data. Real sensor data may require additional filtering steps before being fit for usage for this method.

Future work should also attempt to improve the reliability of the method. Future methods should be able to maintain distances to obstacles at a distance defined by the safety radius, without the need for an additional safety distance. Additionally, future implementations could evaluate other methods for the post-processing step. For example, normal estimation methods could potentially improve the representation of the environment in order to solve the overestimation problem.

Bibliography

[1] Kevin S. Pratt, Robin Murphy, Sam Stover, and Chandler Griffin. Conops and autonomy recommendations for vtol small unmanned aerial system based on hurricane katrina operations. *Journal of Field Robotics*, 26(8):636–650, 2009.

[2] Aleksandar Rodić and Gyula Mester. The modeling and simulation of an autonomous quad-rotor microcopter in a virtual outdoor scenario. *Acta Polytechnica Hungarica*, 8, 2011.

[3] Samir Bouabdallah and Roland Siegwart. Backstepping and sliding-mode techniques applied to an indoor micro quadrotor. In *In Proceedings of IEEE Int. Conf. on Robotics and Automation*, pages 2247–2252, 2005.

[4] Samir Bouabdallah and Roland Siegwart. Full control of a quadrotor. In *Proc. of the 2007 IEEE/RSJ Int. Conference on Intelligent Robots and Systems, Oct 29 - Nov 2*, 2007.

[5] Robert Mahoney, Vijay Kumar, and Peter Corke. Multirotor aerial vehicles: Modeling, estimation, and control of quadrotor. *IEEE Robotics & Automation Magazine*, 19(3):20–32, 2012.

[6] Qingbo Geng and Qiong Hu Huan Shuai. Obstacle avoidance approaches for quadrotor uav based on backstepping technique. In *25th Chinese Control and Decision Conference*, pages 3613–3617, 2013.

[7] Jongki Moon and J.V.R. Prasad. Minimum-time approach to obstacle avoidance constrained by envelope protection for autonomous uavs. *Mechatronics*, 21(5):861—875, 2011.

[8] Oussama Khatib. Real-time obstacle avoidance for manipulators and mobile robots. *The International Journal of Robotics Research*, 5(1):90–98, 1986.

[9] Johann Borenstein and Y. Koren. High-speed obstacle avoidance for mobile robots. In *Proceedings IEEE International Symposium on Intelligent Control*, pages 382–384, 1988.

[10] Johann Borenstein and Y. Koren. Potential field methods and their inherent limitations for mobile robot navigation. In *In Proceedings of the IEEE Conference on Robotics and Automation*, pages 1398–1404, 1991.

[11] Johann Borenstein and Y. Koren. Real-time obstacle avoidance for fast mobile robots in cluttered environments. In *Proceedings of the IEEE International Conference on Robotics and Automation*, 1990.

[12] Johann Borenstein and Y. Koren. The vector field histogram - fast obstacle avoidance for mobile robots. *IEEE Journal of Robotics and Automation*, 7:278–288, 1991.

[13] Johann Borenstein and Iwan Ulrich. Vfh+: Reliable obstacle avoidance for fast mobile robotics. In *In Proceedings of the International Conference on Robotics and Automation*, volume 2, pages 1572–1577, 1998.

[14] Johann Borenstein and Iwan Ulrich. Vfh*: Local obstacle avoidance with look-ahead verification. In *In Proceedings of the IEEE International Conference on Robotics and Automation*, volume 3, pages 2505–2511, 2000.

[15] W. Feiten, R. Bauer, and G. Lawitzky. Robust obstacle avoidance in unknown and cramped environments. In *In Proceedings of the IEEE International Conference on Robotics and Automation*, pages 2412–2417, 1994.

[16] Dieter Fox, Wolfram Burgard, and Sebastian Thrun. The dynamic window approach to collision avoidance. *IEEE Robotics & Automation Magazine*, 4(1):23–33, 1997.

[17] Javier Minguez and Luis Montano. Nearness diagram navigation (nd): A new real time collision avoidance approach. In *In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2094–2100, 2000.

[18] Javier Minguez and Luis Montano. Nearness diagram (nd) navigation: Collision avoidance in troublesome scenarios. *IEEE Transactions on Robotics and Automation*, 20:45–59, 2004.

[19] Javier Minguez. The obstacle-restriction method (orm) for robot obstacle avoidance in difficult environments. In *In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2284–2290, 2005.

[20] Daniel Vikenmark. The obstacle-restriction method (orm) for reactive obstacle avoidance in difficult scenarios in three-dimensional workspaces. Master's thesis, Royal Institute of Technology (KTH), 2006.

[21] Reid Simmons. The curvature-velocity method for local obstacle avoidance. In *In Proceedings of the 1996 IEEE International Conference on Robotics and Automation*, pages 3375–3382, 1996.

[22] Oliver Brock and Oussama Khatib. High-speed navigation using the global dynamic window approach. In *In Proceedings IEEE International Conference on Robotics and Automation*, volume 1, pages 341–346, 1999.

[23] Paul Merrell, Dah jye Lee, and Al Beard. Obstacle avoidance for unmanned air vehicles using optical flow probability distributions. *Mobile Robots XVII*, 5609:13–22, 2004.

[24] John Stowers, Michael Hayes, and Andrew Bainbridge-Smith. Biologically inspired uav obstacle avoidance and control using monocular optical flow & divergence templates. In *Proceedings of the 5th International Conference on Automation, Robotics and Applications*, pages 378–383, December 2011.

[25] Haiyang Chao, Yu Gu, and Marcello Napolitano. A survey of optical flow techniques for uav navigation applications. In *International Conference on Unmanned Aircraft Systems*, pages 710–716, Grand Hyatt Atlanta, Atlanta, GA, May 2013.

[26] Laurent Muratet, Stéphane Doncieux, and Jean arcady Meyer. A biomimetic reactive navigation system using the optical flow for a rotary-wing uav in urban environment. In *Proceedings of ISR*, 2004.

[27] Laurent Muratet, Stephane Doncieux, Yves Briere, Jean arcady Meyer A, and Place Emile Blouin. A contribution to vision-based autonomous helicopter flight in urban environments. *Robotics and Autonomous Systems*, 50:195–209, 2005.

[28] Dong-Wan Yoo, Dae-Yeon Won, and Min-Jea Tah. Optical flow based collision avoidance of multi-rotor uavs in urban environments. *International Journal of Aeronautical & Space Science*, 12(3):252–259, 2011.

[29] Stefan Hrabar and Gaurav S. Sukhatme. Combined optic-flow and stereo-based navigation of urban canyons for a uav. In *In Proceedings of the IEEE/RSJ Int. Conference on Intelligent Robots and Systems*, pages 2–6, 2005.

[30] Stefan Hrabar and Gaurav S. Sukhatme. A comparison of two camera configurations for optic-flow based navigation of a uav through urban canyons. In *In EEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2673–2680. Press, 2004.

[31] Matthias Nieuwenhuisen, Mark Schadler, and Sven Behnke. Predictive potential field-based collision avoidance for multicopters. In *In: Proc. of the 2nd Conference on Unmanned Aerial Vehicles in Geomatics (UAV-g*, 2013.

[32] Nils Gageik, Paul Benz, and Sergio Montenegro. Obstacle detection and collision avoidance for a uav with complementary low-cost sensors. *IEEE Access*, 3:599–609, 2015.

[33] Sebastian Scherer, Sanjiv Singh, Lyle Chamberlain, and Srikanth Saripalli. Flying fast and low among obstacles. In *In International Conference on Robotics and Automation (ICRA*, page 2023, 2007.

[34] João Mendes and Rodrigo Ventura. Safe teleoperation of a quadrotor using fastslam. In *IEEE International Symposium on Safety, Security, and Rescue Robotics*, pages 1–6, November 2012.

[35] Changhong Fu, Miguel A. Olivares-Mendez, Pascual Campoy, and Ramon Suarez-Fernandez. Uas see-and-avoid strategy using a fuzzy logic controller optimized by cross-entropy in scaling factors and membership functions. In *International Conference on Unmanned Aircraft Systems*, pages 532–541, May 2013.

[36] Miguel A. Olivares-Mendez, Pascual Campoy, Ignacio Mellado-Bataller, and Luis Mejias. See-and-avoid quadcopter using fuzzy control optimized by cross-entropy. In *IEEE International Conference on Fuzzy Systems*, pages 1–7, June 2012.

[37] Changhong Fu, Miguel A. Olivares-Mendez, Ramon Suarez-Fernandez, and Pascual Campoy. Monocular visual-inertial slam-based collision avoidance strategy for fail-safe uav using fuzzy logic controllers. *Journal of Intelligent and Robotic Systems*, 73(1-4):513(21), January 2014.

[38] Allen Ferrick, Jesse Fish, Edward Venator, and Gregory S. Lee. Uav obstacle avoidance using image processing techniques. In *IEEE International Conference on Technologies for Practical Robot Applications*, pages 73–78, 2012.

[39] Felipe Patino Vista IV, Ansu Man Singh, Deok-Jin Lee, and Kil To Chong. Design convergent dynamic window approach for quadrotor navigation. *International Journal of Precision Engineering and Manufacturing*, 15, No. 10:2177–2184, October 2014.

[40] Kwangjin Yang, Seng Keat Gan, and Salah Sukkarieh. A gaussian process-based rrt planner for the exploration of an unknown and cluttered environment with a uav. *Advanced Robotics*, 27(6):431–443, 2013.

[41] Kourosh Khoshelham and Sander Oude Elberink. Accuracy and resolution of kinect depth data for indoor mapping applications. *Sensors (Basel)*, 12(2):1437–1454, 2012.

[42] US Air Force. The us air force remotely piloted aircraft and unmanned aerial vehicles strategic vision. *U.S. Air Force Research*, 2005.

[43] Larry Friese, N.R. Jenzen-Jones, and Michael Smallwood. *Emerging Unmanned Threats: The use of commercially-available UAVs by armed non-state actors.* Armament Research Services (ARES), 2016.

Appendix A

Sustainability, Ethics and Social Impact

There are ethical issues regarding both obstacle avoidance and the UAV platform that ought to be taken into consideration.

One of the long-term goals of obstacle avoidance methods are to allow platforms to become autonomous. However, it is implausible to expect that some obstacle avoidance algorithm will have its intended effect in all situations; after all, we do not expect perfect behaviour from human drivers. This can lead to situations where obstacle avoidance fails, and collisions are inevitable. A collision can lead to property and personal damage, which invariably leads to the question of responsibility, as the responsible part is expected to offer compensation. There are a number of identifiable potential culprits in the event of incidents of this type: the implementer of the obstacle avoidance system; the manufacturer of the autonomous platform; the company currently in possession of the platform; an outside third party; or outside factors which could not be taken into account. The legal system may not yet be clear on the subject and as such needs to be addressed.

As mentioned, UAVs have found a range of uses in the public sphere. UAVs were used for inspection of damaged structures in the aftermath of hurricane Katrina, where entering damaged buildings could have been potentially fatal. UAVs have been used for search and rescue operations and for surveillance by authorities, such as the U.S. Air Force[42]. The ability of the UAV for reconnaissance is not immune to nefarious use, however. Fascist governments could use UAVs as a tool to spy on the opposition. However, due to the relatively low cost of producing UAVs, the barrier for ownership is low. Many people have the resources to purchase a UAV and use them to their liking. UAVs are employed by enemy insurgents such as the Islamic State[43]. Regulations of UAVs may be imposed in the future.

The purpose of developing obstacle avoidance methods is generally to work towards automation. The local obstacle avoidance method of this thesis and similar methods could be integrated into a higher level path planning algorithm which could allow for UAVs to navigate and carry out tasks without human supervision. In a commercial setting, this may lead to significant salary cuts, or even layoffs, due to the decreased demand on skilled UAV pilots that automation will invariably bring.

Automation has historically had a disruptive effect on the job market where human worker simply cannot compete with machines which are more efficient, does not require any OSHA compliance and cannot join unions. If a worker can be replaced by a machine in a free market, there is little incentive to not do so. However, new technologies also tend to create new jobs previously unheard of. The work on automation will most likely lead to a future with less manual labour and more cognitively demanding work, at least until such processes can be automated as well.

Sustainability is not directly relevant to obstacle avoidance algorithms. Instead the sustainability aspect appears with the hardware of the platform. A UAV can be composed of many different materials, but one popular candidate material is plastic. Given the increasing consumer demand for UAVs for both professional and recreational tools, it can be assumed that the production of UAVs contributes to the production of plastic. Also, since UAVs are electronic products, there is a chance that they contain "conflict minerals", which are typically extracted in countries which are guilty of systematic human rights abuses such as the Democratic Republic of the Congo. Consumption of consumer UAVs may therefore indirectly fund humanitarian atrocities. At the end of the product's life cycle, it is probable that UAV platforms, potentially rich in plastic and hazardous metals, are buried in landfills.