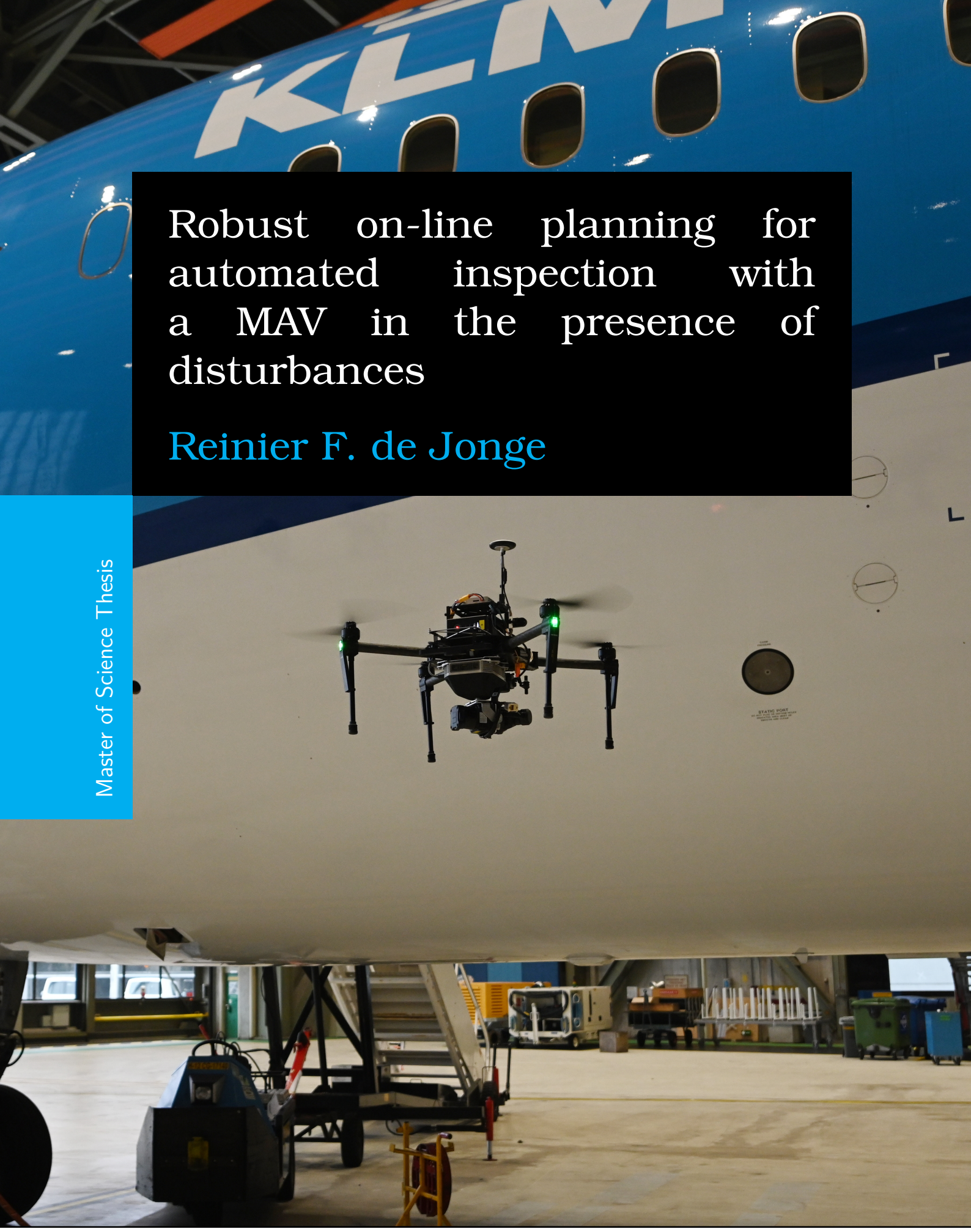
Robust on-line planning for automated inspection with a MAV in the presence of disturbances

Reinier F. de Jonge

Master of Science Thesis



TU Delft Delft University of Technology

Delft Center for Systems and Control

Robust on-line planning for automated inspection with a MAV in the presence of disturbances

Master of Science Thesis

For the degree of Master of Science in Systems and Control at Delft University of Technology

Reinier F. de Jonge

July 31, 2019

CONFIDENTIAL

Faculty of Mechanical, Maritime and Materials Engineering (3mE) · Delft University of Technology

The work in this thesis was supported by Mainblades Inspections. Their cooperation is hereby gratefully acknowledged.

Abstract

Safe and reliable autonomous inspection tasks using Micro Aerial Vehicles (MAVs) in cluttered environments are challenging due uncertainties encountered during inspections. In general, algorithms consist of pre-planning paths and executing them. These precomputed inspection paths do not consider potential occlusion by obstacles. In turn, this could render the path infeasible and result in an incomplete inspection of the object to inspect. In order to solve this, a robust method is required, defined as the mitigation of viewing disturbances. This thesis presents an on-line inspection method for occluded environments with static obstacles.

The proposed method splits an off-line computed global inspection path into segments and treats each segment as a separate inspection problem. By using an information-based cost function, an Model Predictive Controller (MPC) allows for robust on-line inspection of each of these segments by rejecting obstacle occlusion and collision, if necessary. Additionally, the cost function is designed to be submodular, a mathematical property describing diminishing returns and allowing greedy optimisation while obtaining performance guarantees.

Properties of the method are demonstrated in two different environments: with and without obstacles. Due to many sigmoid functions within the cost function, it is a complex optimisation problem. For this reason, scalability is tested and measured in amount of triangles to determine feasible-sized inspection segments. It is shown that for 16 triangles with one obstacle (both of arbitrary sizes), the calculation time for each MPC iteration starts to exceed $15Hz$, becoming more unstable with each added triangle. This effect can be mitigated by discarding information cost of the intermediate cost function.

Finally, it is shown that, where the global inspection path is partly occluded, the proposed method handles occlusion by obstacles during inspection at the cost of increased duration of the inspection, while maintaining quality of the solution. However, the predicted performance guarantee by submodularity does not always hold in practice. Future work can explore the integration of sensor measurements to the method or use learning-based approaches to reduce the required on-line computational resources.

Table of Contents

List of Figures

List of Tables

Preface

The work presented in this report is part of the thesis project in the scope of the Master of Science program Systems and Control of the Mechanical, Maritime and Materials Engineering (3mE) faculty.

The project has been proposed by the founders of Mainblades Inspections B.V., Ir. J. Verboom and Ir. D. Borotaİn collaboration with Dr. J. Alonso-Mora , Assistant Professor at the Cognitive Robotics department, the assignment was formulated and approved for a student following his Masters at Delft Center for Systems and Control (DCSC).

The final committee consists of the following people:

- Prof.dr.ir. J. Hellendoorn
- Dr. J. Alonso-Mora
- Ir. H. Zhu
- Ir. J. Verboom
- Ir. D. Borota

I would like to thank friends, family, colleagues and supervisors of the TU for aiding me through the graduation process, be it providing support, feedback, confidence, criticism, etc. They have helped me a lot over the past year and are the reason this work came to its current state.

Delft, University of Technology
July 31, 2019

Reinier F. de Jonge

Chapter 1

Introduction

The popularity of Micro Aerial Vehicles (MAVs) has increased over the past years, allowing research and the commercial market to flourish. Greater computational power motivates MAVs to be deployed in increasingly complex, but also potentially hazardous, problems. Hence, the popularity will most likely keep rising in the future years, following the trend as shown in Figure 1-1.

MAVs lend themselves to be excellent tools for inspection, as they are small, agile, lightweight and potentially remove the need to use humans in hazardous tasks. Automated inspection performed with robots is an interesting task with many applications. For instance, it can be used to detect damage of aircraft or off-shore structures, but also potential intruders of land areas. Performing autonomous inspection requires the robot to make sense of its surroundings and be able to interpret the object to inspect in order to determine the course of action. Therefore, it is a complex problem and many factors need to be accounted for before a working, reliable solution is achieved. The complexity increases radically when the object to inspect is not what it is expected to be, it is a moving system of there are obstacles unaccounted for. Developing an algorithm for an MAV that is able to quickly generate a new plan when obstacles are present is a challenging task and the goal of this thesis.

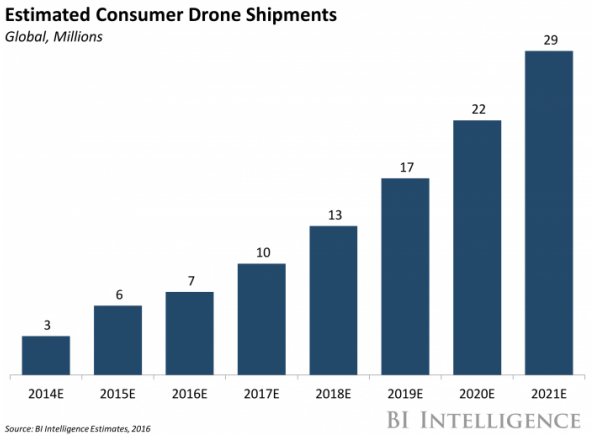


Figure 1-1: A diagram showing the estimated global MAV shipments[1]

[1] Image downloaded from http://www.dronesglobe.com/news/drone-market-share-analysis-predictions-2018/ on October 2018.

1-1 MAV inspections

The use of MAVs for commercial and private purposes has increased over the past few years and inspection of large and complex structures are now within reach. A few reasons as to why it is so popular are mentioned in the following paragraphs and are summarised in Table 1-1.

Risk Inspecting large objects has the potential to become very dangerous, due to their height and therefore, use of proper platforms is required. Weather conditions potentially impose dangerous conditions to perform the inspection properly. Having the ability to deploy an MAV allows people to stay indoors and on the ground.

Duration Manually inspecting objects, with the eye or even with a camera, requires a person to manoeuvre around the object and carefully look at the whole exterior. Depending on the size of the object, and if extra structures are needed to view certain parts of the exterior, the duration of an inspection might increase steeply. An MAV is able to move towards hard-to-reach positions, due to its agility and size, potentially saving a lot of time.

Quality Saving the images or video footage of an autonomous inspection can easily be shared and reported, as it is digital data. Moreover, learning algorithms are able to locate possible imperfections or damages of the object and are not affected by fatigue of people. Even though people might be trained, judgement of inspections can be affected.

Ease of operation Instead of physical labour, performing an inspection with an MAV can potentially be done with the press of a button.

Table 1-1: Summary of a comparison between manual- and MAV inspections.

	Manual inspection	Inspection using MAV
Risk	**High**: the use of extra structures is potentially hazardous.	**Low**: the inspector can stay indoors and on the ground.
Duration	**Long**: the use of extra structures costs a lot of time.	**Short**: hard-to-reach places are easier to reach, saving time.
Quality	**High**: trained inspectors are able to detect imperfections.	**High**: a lot of data generation, use of learning algorithms for detection, not affected by tiredness.
Ease of operation	**Low**: physical labour	**High**: can potentially be done with a single button.

1-2 Technical background

This section will provide technical background regarding the type of MAV used, assumptions and scope of the inspection problem.

1-2-1 What is an inspection?

An inspection is defined to be the traversal of a route where images will be taken at certain locations, called a viewpoint ν. Taking an image I should contribute to covering a percentage of object to inspect B. The images will be taken with a camera that is attached to the MAV and has two degrees of freedom, yaw φ and pitch ψ. Figures 1-2 and 1-3 illustrate this. For future reference, the image plane that is possible to be captured by the camera will be referred to as 'viewing' from a position. When a sequence of images is stacked together, the combined area A_I of B that is covered is considered inspected. The object of interest should be modelled as a mesh, M, and multiple images can be taken at a single viewpoint. The images give information about the state of the surface of B. However, processing images does not fall under the scope of this thesis.

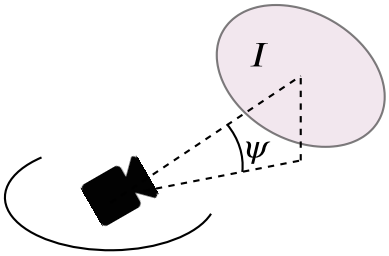


Figure 1-2: Illustrating the yaw φ and pitch ψ to capture image I.

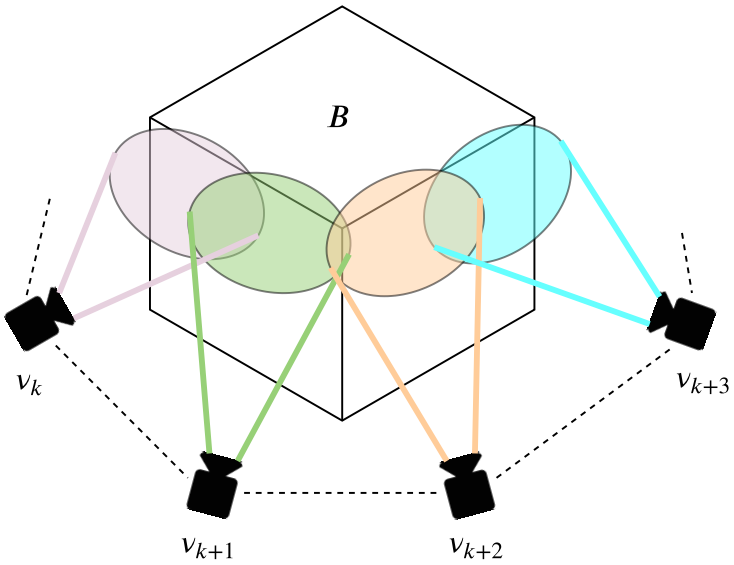


Figure 1-3: A simple illustration where, for each viewpoint ν, a part of object to inspect B is captured. The dotted line represents the inspection path and, when completed, the combined area A_I is considered inspected.

Global requirements It is quite clear that, especially for aircraft inspections, full coverage of the object is desired. A predefined route, as explained in Section 2-1, is used as the global plan and is the guideline for execution of the inspection. This route defines a set of viewpoints

$\mathcal{V}$ that, when images are taken at these locations, cover a certain area A_{gp} of B. Since the global plan is the guideline, A_{gp} becomes the coverage to achieve.

1-2-2 MAV Hardware

A company that inspects objects using MAVs and fits such a description is Mainblades Inspections (MBI). They develop a plug-and-play tool for an MAV, called Intelligent Sensing and Automated Control (ISAAC), enabling autonomous inspection of aircraft using an MAV. ISAAC improves the overall process significantly compared to manually inspecting aircraft and is an add-on for an MAV, consisting of all necessary equipment to inspect aircraft and take pictures of the whole exterior. The type of MAV and its hardware specifications with which the inspection is performed by MBI is shown in Table 1-2.

Table 1-2: Specifications of the used MAV, including ISAAC.

MAV type	DJI Matrice 100[1]
Max. ascend velocity	$5m/s$
Max. descend velocity	$4m/s$
Max. velocity	$22m/s$
Diagonal wheelbase	$0.65m$
Onboard Camera	DJI Zenmuse X5[2] for inspection
External Board	UP Squared[3], Intel Pentium QuadCore 2.5GHz CPU, 8GB LPDDR4
External Sensors	Intel RealSense Depth Camera[4] for navigation
	Rotating LIDAR[5] with 270 deg field of view at $0.5Hz$
	Internal measurement unit

A quite limiting factor for an inspection is, of course, the MAV itself. Depending on the type, size and modifications, the durability, quality, are affected. The following paragraphs will highlight a few assumptions.

Real-time processing Solving the problems in real-time is an excellent feature and required for full on-board navigation. Even though the scope of this thesis does not require real-time processing for the specification stated in Table 1-2, keeping this in mind will benefit the applicability of the presented solution. Therefore, this is a goal constraint.

Camera location The location of the camera is fixed. When hovering, it is directly below the centre of mass by $0.2m$.

Camera field of view The Field of View (FOV) is limited due to the propellers being in sight of the camera. For consistency and simplicity, the same field of view as the Laser Imaging Detection and Ranging (LIDAR), 270 deg, will be considered. This will be classified as a hard constraint.

[1] DJI, https://www.dji.com/matrice100, accessed on November 12th 2018
[2] DJI, https://www.dji.com/zenmuse-x5, accessed on November 12th 2018
[3] Up-board, https://up-shop.org/up-boards/97-up-squared-pentium-quad-core-8gb-memory64gb-emmc.html, accessed on November 12th 2018
[4] Intel, https://click.intel.com/intelr-realsensetm-depth-camera-d435.html, accessed on November 12th 2018
[5] Hokuyo, https://www.hokuyo-aut.jp/search/single.php?serial=170, accessed on November 12th 2018

Thrust The MAV has four propellers to be controlled individually. Depending on the control architecture (see Section 4-1), the control input may be represented in different ways. However, the DJI is not allowed to produce negative thrust and can only generate some theoretical maximum relating to the maximum velocity. This depends on the ascending velocity specified in Table 1-2 and will be a hard constraint of $5m/s$. The combined thrust input will be $50N$ at maximum.

1-2-3 Dynamical model of the MAV

Designing the dynamical model of the MAV is not included in the scope of this thesis. The model will be static and describe the MAV position. It will consist of the following state- and input vectors:

$$\mathbf{x} = \begin{bmatrix} x & y & z & \dot{x} & \dot{y} & \dot{z} \end{bmatrix}^T \tag{1-1}$$

$$\mathbf{u} = \begin{bmatrix} F_{thrust} & \varphi & \theta & \psi \end{bmatrix}^T \tag{1-2}$$

Where x, y, and z are positional parameters and θ, ψ and φ represent the roll, pitch and yaw, respectively. Furthermore, the dynamical model is the same as described by Bouabdallah and Siegwart 2004, except using the Robotic Operating System (ROS) axis convention[2](east-north-up, *Standard Units of Measure and Coordinate Conventions*):

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{bmatrix} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ ((\cos(\psi)\sin(\theta)\cos(\varphi) + \sin(\psi)\sin(\varphi))F_{thrust})/m \\ ((\sin(\psi)\sin(\theta)\cos(\varphi) + \cos(\psi)\sin(\varphi))F_{thrust})/m \\ (\cos(\varphi)\cos(\theta)F_{thrust})/m - g \end{bmatrix} \tag{1-3}$$

1-2-4 Environmental representation

From a world perspective, the problem can be confined to an environment in which the method should operate. This environment contains information on boundaries and infeasible locations, enabling a clear confined problem.

World The workspace, will be defined as a box, varying per inspection. This environment can be cluttered with many obstacles, such as humans, poles or platforms. However, a single object must be appointed to inspect, on which size of the closed environment depends. The workspace is equal to the confining box minus the obstacles and object to inspect. An example is shown in Figure 1-4.

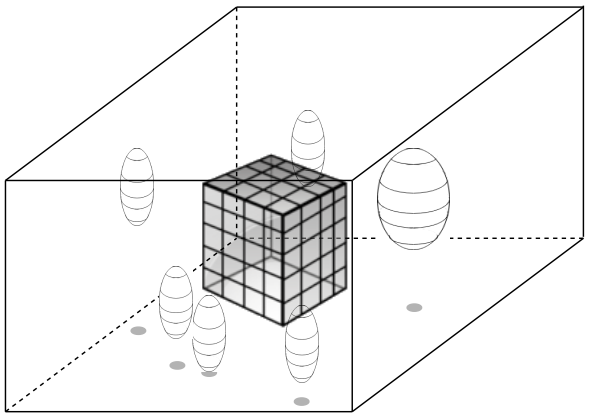


Figure 1-4: A schematised cluttered workspace in which the inspection problem operates.

Object to inspect The object to inspect must be represented as a mesh model, also containing the normals of each triangle. In order for the mesh to be useable with the global planner (Section 2-1), it must be watertight. Furthermore, inspection wise, the user determine the minimum- and maximum allowable distance to view this object.

Obstacles For the scope of this thesis, all obstacles inside the working environment are assumed to be known and represented as ellipsoids of arbitrary size. Reason for using these, is that ellipsoids can easily be manipulated mathematically to benefit the current problem (further explained in Section 4-3-2).

MAV The MAV is represented as a fixed bounding sphere, to make use of the easy manipulation of ellipsoids (Section 4-3-2). Moreover, the true state values are known at any point in time for this thesis.

The previous paragraphs created a set of constraints under which the inspection must operate. Table 1-3 shows a summary of the assumptions and constraints for the scope of this thesis.

Table 1-3: Summary of the boundaries and constraints.

Type	Remark
World	Defined as a box.
Obstacles	Known and represented as ellipsoids.
Object	Represented as a watertight mesh model M. Only one allowed.
Real-time	No hard constraint. Increases overall applicability.
Camera FOV	A 270 degree FOV.
Sates	Upward velocity is constrained to 5 m/s.
Input	Maximum thrust of 50 N.
Radius	Diagonal wheelbase of $0.65m$
Dynamics	The dynamical model is static.

1-3 Problem definition

This section provides the formally defined problem for the scope of this thesis, which the remainder of this document is built upon.

1-3-1 Formal definition

The global inspection path is calculated off-line, providing maximal coverage. However, it only accounts for object to inspect, B. It is possible that an MAV is unable to execute the path correctly due to unforeseen disturbances such as obstacles, occluding B or obstructing viewpoints, resulting in a non-robust solution.

Robustness is not defined in the pure control sense, such that external disturbances (wind, pushes) or uncertain system parameters are rejected, but defined as mitigating viewing disturbances.

Handling viewing disturbances on-line requires a robot to be able to re-plan its course of action, relying on live sensory data. For p viewpoints, the algorithm must, therefore, compute and follow a path to inspect B and plan locally to mitigate observed obstacles. The difficulty lies within correcting for these disturbances that influence the behaviour in order to correctly complete the inspection.

Robustness will be achieved by locally correcting for obstacles, both in terms of collisions and occlusion, while keeping the same, or better, performance than the off-line calculated global inspection path. In other words, when no disturbances are encountered, the global path results in maximal coverage.

1-3-2 Mathematical definition

In the previous sections, a high level background is given. The following paragraphs give a mathematical definition of the necessary background.

Environment Consider a bounded semi-known environment $\mathcal{W} \subset \mathbb{R}^3$, containing rigid mesh model M and obstacle set $\mathcal{O}$.

M is comprised of n triangles. Triangle T_i consists of three points $\mathbf{A}, \mathbf{B}, \mathbf{C} \in \mathbb{R}^3$ and a normalised vector $\mathbf{n} \in \mathbb{R}^3$ describing the direction, positive when facing away from the surface.

Obstacles are static and represented as ellipsoids, of which the general equation is given by:

$$\frac{x^2}{a^2} + \frac{y^2}{b^2} + \frac{z^2}{c^2} = 1 \tag{1-4}$$

Here, a, b and c are known as the radii of the ellipsoid: $\mathbf{r}_O = \begin{bmatrix} a & b & c \end{bmatrix}^T$. The centre location is denoted by $\mathbf{O}_c \in \mathbb{R}^3$.

Observability In order to view, or observe, triangle $_i$, two criteria must be met:

1. $\gamma \leq \delta$ s.t. $\gamma = \angle \mathbf{P}\mathbf{t}\mathbf{n}_i$
2. $d_{min} \leq \|\mathbf{P} - \mathbf{t}\| \leq d_{max}$

Where γ is the incidence angle, $\mathbf{P} \in \mathbb{R}^3$ the camera position, $\mathbf{t} \in T_i$ and δ, d_{min} and d_{max} are user defined variables stating the maximum allowable incidence angle, minimum- and maximum distance to any T, respectively. Figure 1-5 illustrates a triangle and the incidence angle.

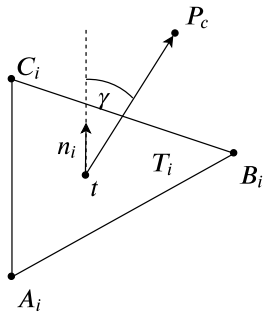


Figure 1-5: A schematised triangle illustrating the incidence angle.

Global path Given is a global, dynamically feasible, path ξ, connecting an ordered set of viewpoints $\nu_k \in \mathcal{V}$ for $k = 1, \ldots, p$. More specifically, the path is a straight line between viewpoints and defined as $\xi_{k-1}^k = (1-s)\nu_{k-1} + s\nu_k$ with $s \in [0, 1]$.

Each viewpoint ν_k is viewing a subset $m_k \subseteq M$ triangles, such that $\bigcup_{k=1}^{p} m_k = M_{gp}$, where M_{gp} equals all the triangles viewed by the viewpoints of global path ξ.

When there are no disturbances around B, taking images at $\mathcal{V}$ completes the inspection and observes all triangles $T \in M_{gp}$. The subset $\{M_{gp} \setminus M\}$ is not considered in the scope of this thesis. When disturbances are present, it is possible that the set of inspected triangles $m_{inspected} \neq M_{gp}$. The remainder subset $m_{infeasible} = \{m_{inspected} \setminus M_{gp}\}$ is defined as the infeasible set of triangles.

1-3-3 Thesis approach

To find a solution to the problem defined in the previous section, the approach in Figure 1-6 will be followed to reach the main objective:

> *"Design an on-line algorithm that is able to perform visual inspection of a 3D mesh object, collision-free in an occluded environment, implementing a Model Predictive Controller and guaranteeing full coverage."*

The thesis is divided into three parts. Chapters 2 and 3 focus on proposing the used methods and compares current limitations of state-of-the-art work; Chapters 4 and 5 propose the method and motivate choices made that lead to the algorithm; Chapters 6 and 7 evaluate experimental results to verify the proposed solution.

Contributions

The contributions made by this thesis can be listed for the academic world and the industry and are formulated as follows:

- Automated inspection using MAV:
  - using MPC as an on-line inspection method;
  - robust on-line inspection by avoiding collisions- and occlusion of obstacles;
  - guaranteeing the same or better performance than the precomputed global path.
- Mainblades Inspections:
  - developing a control framework of the MAV;
  - avoiding collisions while performing inspections;
  - implementing suitable methods applicable to the MAV used for inspections;

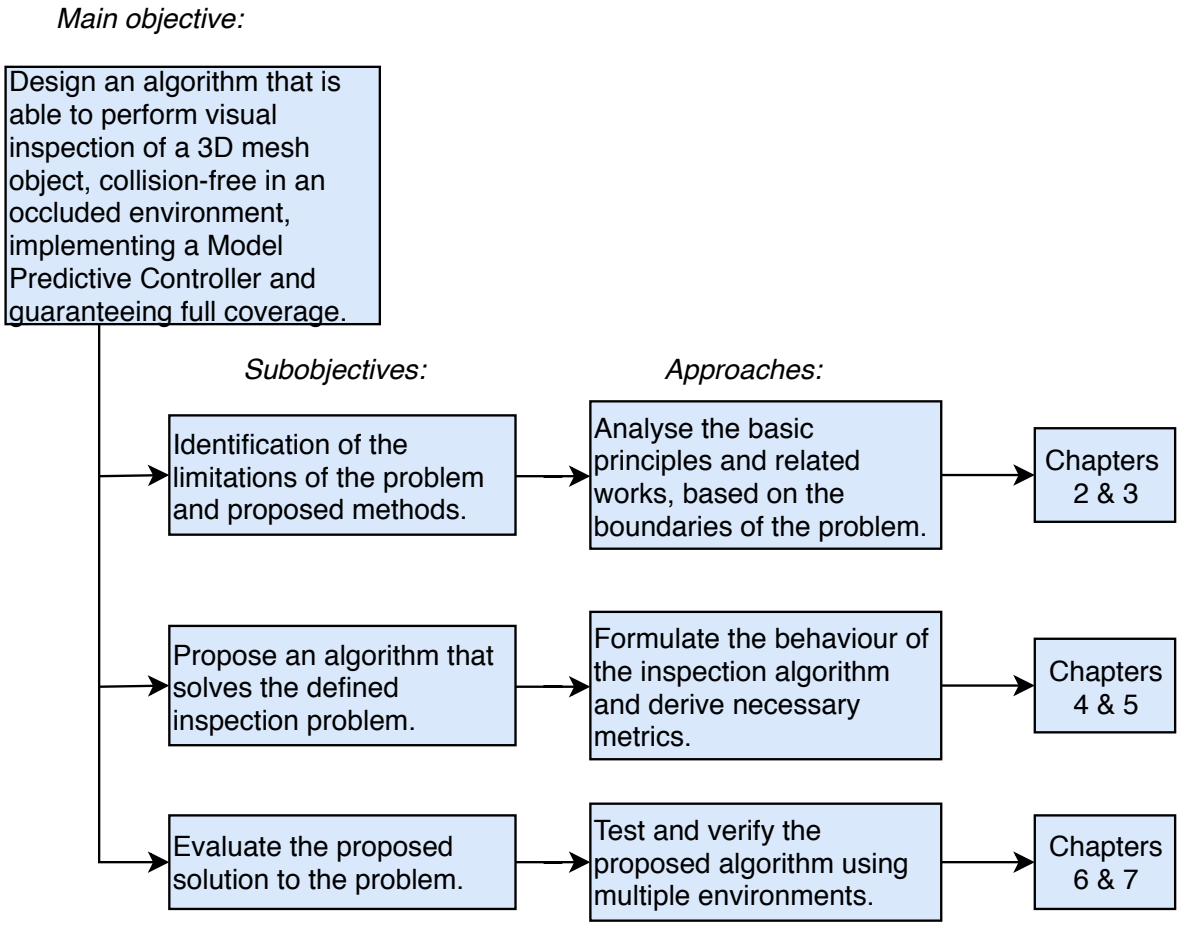


Figure 1-6: Structure of the thesis.

Chapter 2

Preliminaries

This chapter aims to introduce the concepts necessary for the proposed method. Analysing these concepts provides insight contributing to the final method, proposed in Chapter 4.

The related works are a great tool to build upon and provide building blocks for the eventual solution. First, the view global planner algorithm will be discussed, which generates an initial inspection tour. Then, the mathematical property submodularity will be explained with its application to inspections. Lastly, a brief introduction to MPC is given.

2-1 View global planner

The global viewpoint planner (Potdar 2018, found in appendix A) is an algorithm which generates a path of viewpoints that each cover a certain area of the input model, as well as satisfy certain photogrammetry constraints. It is an off-line algorithm and provides the basis of the inspection tour. This section gives a short summary as it is used as the basis for the proposed method.

2-1-1 Input of the algorithm

Using a set of inputs, the algorithm is able to generate a route to be used for inspection. Using the same object, different input may result in very different routes. The following are needed to obtain a certain output.

Camera characteristics In order to be able to determine what the camera is able to see, a model of it is required, the viewing frustum. One is defined to be the region of space

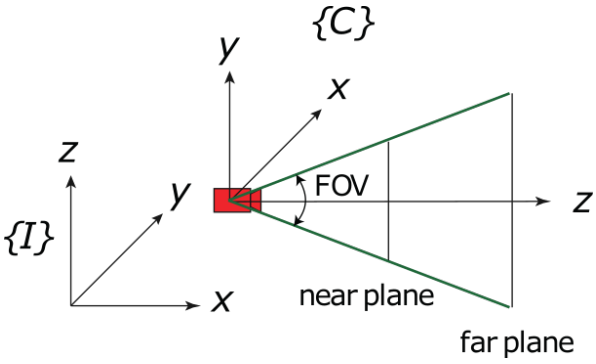


Figure 2-1: Camera frustum for global viewpoints planner. Here, $\{I\}$ is the world reference frame and $\{C\}$ the frame of the robot. Source: Potdar 2018

that appears on the screen of the camera. For this algorithm, the shape is defined as rectangular, even though the shape might vary on the type of camera lens that is used. Figure 2-1 shows such a viewing frustum and the parameters that are used. The Field of View (FOV) is the angle at which the camera is able to see in vertical direction. The horizontal direction is determined by an aspect ratio. The near- and far planes denote the minimum and maximum distance clear images for inspections can be taken from. It can also be considered as a high-level constraint.

Mesh model The object to inspect must be represented as a mesh model (see Figure 2-2), consisting of triangles and normals, similar to Section 1-3. These models can easily be found on websites or made by existing algorithms, for example by Marton *et al.* 2009 who create a mesh based on a point cloud. For the algorithm to work, surface normals for each triangle must be present as it must be determined from which side the triangle can be viewed.

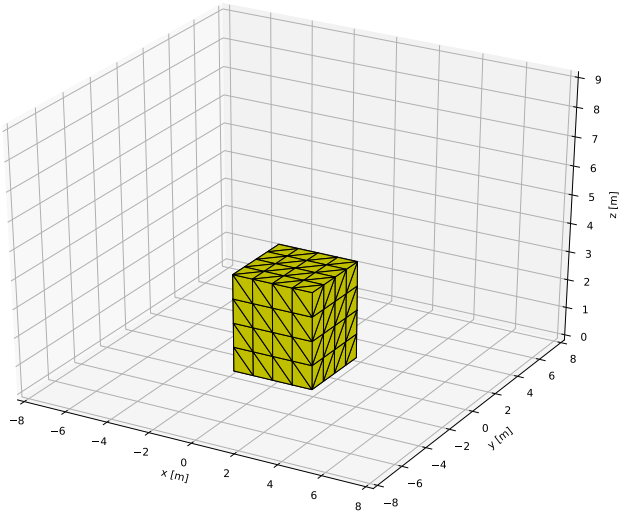


Figure 2-2: An example mesh model representation of a cube.

Constraints and requirements A few parameters and constraints can be configured by the user as input of the global planner. These constraints will influence the overall behaviour and, of course, the output. The configuration consists of

- the size of the workspace;
- cubic cell size for grid sampling;
- vehicle (motion) constraints;
- a maximum incidence angle;
- a minimum number of views for each triangle;
- complete projection of each triangle on an image plane.

Here, the vehicle constraints also contain the allowed pitch and yaw angles, the incidence angle shares the same definition as α from Section 1-3, the image plane follows from the camera characteristics from this section and sampling results in a cellular decomposition in the form of a uniform Cartesian grid.

2-1-2 Methods and output

The algorithm can be summarised in a few steps, namely generating viewpoints, filtering viewpoints and solving for obtaining the final inspection tour.

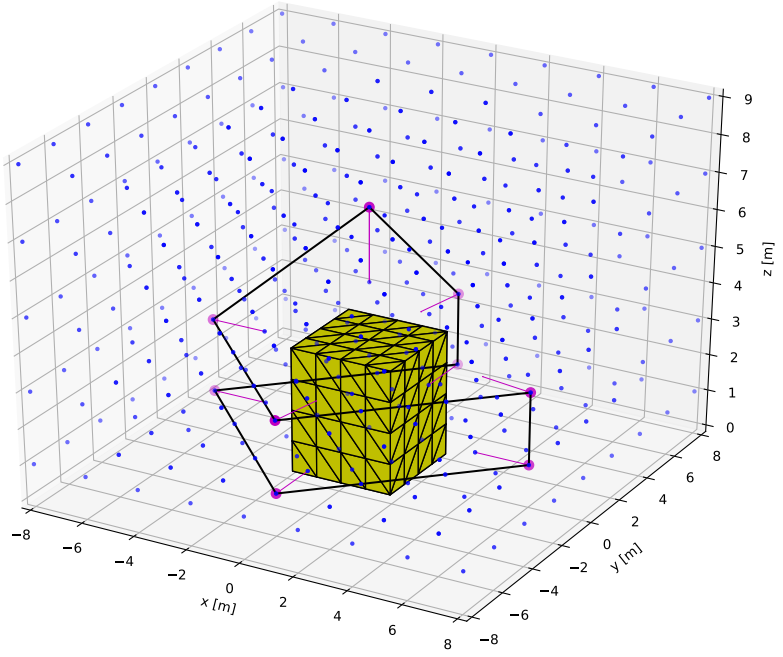


Figure 2-3: The generated tour by the view global planner algorithm is illustrated as a black line, connecting the chosen purple viewpoints. The blue dots represent the sampled potential viewpoints.

Generating viewpoints The generation of viewpoints is depends on two parameters, the confined world in which it samples and the cubic cell edge length. Using these, potential viewpoints will be sampled in a uniform grid, spacing exactly the specified edge length. Each also contains multiple orientations of the camera. An example can be found in Figure 2-3, where the blue dots represent the potential viewpoints for the next step of the algorithm.

Parameters should be carefully chosen. The world is straightforward, as it must be large enough such that it contains the complete mesh. The edge length, however, not so much. A smaller edge length means higher resolution and a better solution. However, it increases the computational load potentially to the third power.

Filtering and grading viewpoints Filtering viewpoints is performed in two steps. First, the proximity of a potential viewpoint with respect to the mesh is checked against the minimum- and maximum distance. It is then removed if the requirement is not met.

Second, ray casting is performed to check each orientation yielding an amount of viewed triangles. Ray casting casts a single ray for each pixel of the image plane and checks if it collides with the mesh, see Figure 2-4. Moreover, on-line planning should avoid this procedure, as it is computationally very expensive. Hence, the pre-filtering of viewpoints based on location.

This step allows for precise measurement of viewed triangles, grading each viewpoint based on location, orientation and view.

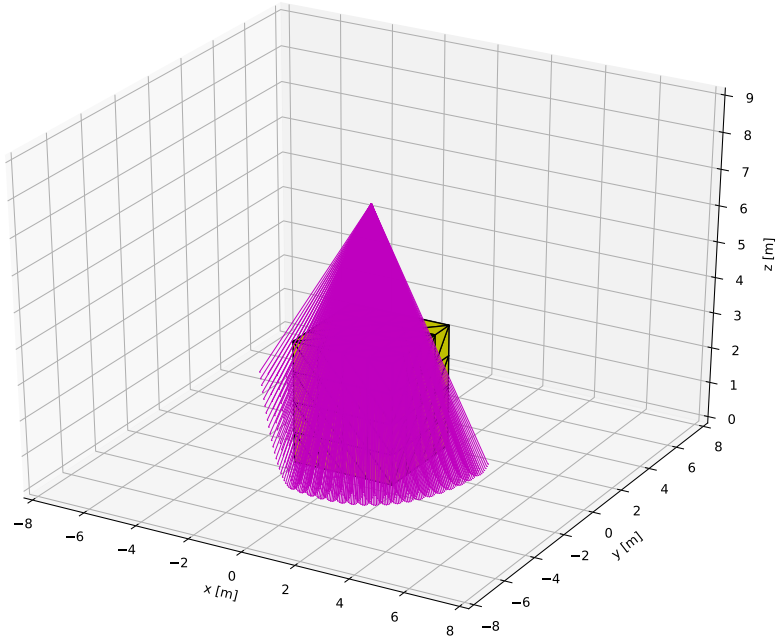


Figure 2-4: Rays cast from a potential viewpoint for the ray casting procedure.

Optimising the tour The final tour is obtained by solving an optimisation problem, specifically the Travelling Salesman Problem (TSP) using the Lin-Kernighan Heuristic and the Google Optimization Toolbox (Potdar 2018). Nodes are greedily picked by the most triangles viewed of the object. From these viewpoints, a tour is generated as short as possible such that every triangle is viewed. An example is shown in Figure 2-3.

Output The optimisation step results in the global inspection path, which is dynamically feasible, viewing as much of the mesh as possible. Besides the route, subset $M_{gp} \subseteq M$ is obtained, containing all observable triangles that the MAV is able to observe. The limiting factor is the gimbal camera, which restricts viewing the mesh from underneath and causes unobservable triangles.

2-2 Submodularity

A mathematical property describing diminishing returns or in other words, diminishing marginal gain, is called submodularity. This is particularly useful for problems where information must be gained and is viewed as the discrete version of convexity. It exhibits preservation of submodularity under several mathematical operations, such as taking non-negative linear combinations, residuals and truncation (Nemhauser *et al.* 1978). Even though constrained maximisation of submodular functions is NP-hard, heuristics provide great approximations to such problems. First, set functions are discussed, since submodularity is based on these, followed by submodular theory and approaches.

2-2-1 Set functions

Normally, functions require a single input parameter to evaluate at. Set functions, however, require an input set which is a collection of distinct objects. An example of an input set, is a

set of real numbers, locations or in this case, it would be triangles of a mesh. Let the ground set be the set of triangles to inspect:

$$M_{gp} = \{m_1, m_2, \ldots, m_n\} \tag{2-1}$$

Then, the power set (all possible combinations of a set) of M_{gp} is

$$2^{M_{gp}} = \{\{\}, \{m_1\}, \{m_2\}, \{m_1, m_2\}, \ldots, \{m_1, \ldots, m_n\}\} \tag{2-2}$$

where $\{\}$ is the empty set $\emptyset$. The amount of possible combinations is equal to $2^{|\Omega|}$, where $|\Omega|$ is the cardinality of set M_{gp} and it follows that the domain of a set function is equal to its power set. Formally, a set function should return a real number and is defined as follows:

$$f : 2^{M_{gp}} \rightarrow \mathbb{R} \tag{2-3}$$

By definition, evaluating the empty set must return 0.

2-2-2 Submodular theory

This subsection discusses the theory, followed by an example, different types of submodularity and its application for this problem. The setup illustrated in Figure 2-5 is used as a lead to explain the theory.

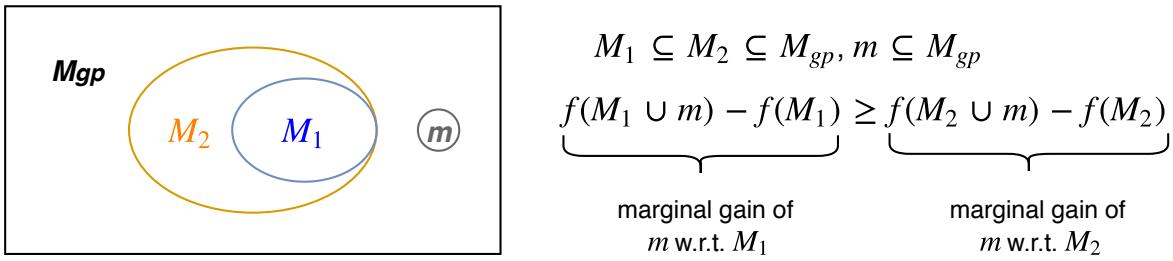


Figure 2-5: Submodular example for Equation 2-6.

Theory Suppose there are two viewpoints, ν_1 and ν_2, which translate to a certain set of meshes M_1 and M_2 to be viewed from that position, respectively. Both M_1 and M_2 are subsets of the ground set M_{gp} and M_1 is a subset of M_2. Also, there is a new viewpoint ν_3, corresponding to set m and subset of M_{gp}. In mathematical terms:

$$M_1 \subseteq M_2 \subseteq M_{gp} \ , \ m \subseteq M_{gp} \tag{2-4}$$

Evaluating the set function f using M_1, M_2 or m as input yields a real number value. The diminishing returns property means that adding m to either M_1 or M_2 yields a higher marginal gain for the smaller set than for the larger set. Mathematically, the marginal gains for set functions are defined as:

$$\text{gain} = f(M_1 \cup m) - f(M_1) \tag{2-5}$$

Then, if the following equation holds for any pair of sets as defined in 2-4, the set function f holds the submodular property:

$$\begin{aligned} &\forall M_1 \subseteq M_2 \subseteq M_2 \cup m, \\ &f(M_1 \cup m) - f(M_1) \geq f(M_2 \cup m) - f(M_2) \end{aligned} \tag{2-6}$$

The submodular condition can be written in a different, more general way for any two subsets of M_{gp}:

$$\begin{aligned} &\forall M_1, M_2 \subseteq M_{gp}, \\ &f(M_1) + f(M_2) \geq f(M_1 \cup M_2) + f(M_1 \cap M_2) \end{aligned} \tag{2-7}$$

Example Using a simple example in the case of Figure 2-5. Let $M_1 = \{m_1, \ldots, m_5\}$, $M_2 = \{m_1, \ldots, m_{10}\}$ and $m = \{m_{11}\}$ and f returns the amount of triangles contained in the mesh. In other words, $f(M) = \sum_{i \in M} w_i$, where $w_i = 1 \forall i$. Since m is located outside both sets M_1 and M_2, Equation 2-6 becomes:

$$\begin{gathered} f(\{m_1, \ldots, m_5, m_{11}\}) - f(\{m_1, \ldots, m_5\}) \geq f(\{m_1, \ldots, m_{10}, m_{11}\}) - f(\{m_1, \ldots, m_{10}\}) \\ 6 - 5 \geq 11 - 10 \end{gathered} \tag{2-8}$$

This holds true due to the equality sign. However, this was the case where m is located outside of M_1 and M_2. Other possibilities are that m is located inside either M_1 or M_2, or perhaps both, as indicated in Figure 2-6.

Looking at the first case, using $m = \{m_7\}$ as example, the union $M_2 \cup m = M_2$ and Equation 2-6 becomes:

$$\begin{gathered} f(\{m_1, \ldots, m_5, m_7\}) - f(\{m_1, \ldots, m_5\}) \geq f(\{m_1, \ldots, m_{10}\}) - f(\{m_1, \ldots, m_{10}\}) \\ 6 - 5 \geq 10 - 10 \end{gathered} \tag{2-9}$$

This still holds, now due to the inequality sign. If $m = \{m_1\}$, it is clear that both gains become zero, as m already exists in both sets. Since these three possible scenarios all hold, the function f is submodular. Lastly, an example where all scenarios are combined would be one where $m = \{m_1, m_7, m_{11}\}$. The gain w.r.t. M_1 is 2, whereas the gain w.r.t M_2 is 1. This is a clear example of where the diminishing returns come into play and that Equation 2-7 is a mathematical way of showing the diminishing returns property.

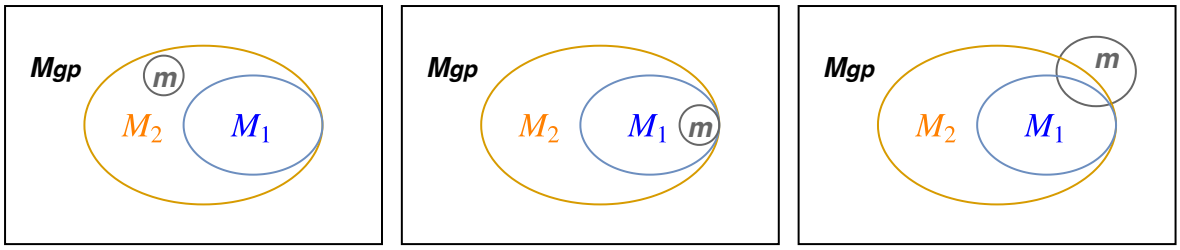


Figure 2-6: Submodular examples for difference cases of m.

2-2-3 Application

Since the problem states it follows a route of viewpoints, defining a submodular function, with an input as described in Subsection 2-2, a relation between the information of the mesh and location of the camera is established. What follows, is that, for a sequence of viewpoints, the maximum amount of information for a position is desired.

The key insight of submodular functions lies in its ease to optimise. Constrained submodular optimisation is an NP-hard problem, but can be approximated by a greedy approach. Moreover, the best possible approach to these kinds of problems is a greedy one (Krause and Golovin 2014) and comes within a factor $(1 - e^{-1})$ of the optimal solution. See Appendix B for proof of the greedy method. In the case of on-line inspection, finding the next location to go to generally is the problem to solve. In itself, it already is a form of a greedy optimisation. Moreover, when the defined submodular problem reflects an inspection gain metric, then solving this locally means that the discretisation by the view global planner becomes overruled and improves quality of viewpoints.

2-3 MPC basics

Predicting a system its behaviour over time allows a user to calculate a trajectory and act on future events. That is, changing the predicted course due to an obstacle or deviation from an objective. Model predictive control uses this philosophy and optimises input resulting in a trajectory, given a certain objective function and constraints. Moreover, it optimises a trajectory which has a length of a user defined time horizon.

The control cycle of an MPC is applying a single input for the discrete time step of the model. Then, the new state of the system is measured and the optimisation problem solved again to obtain new optimal input. This way, MPC becomes an on-line procedure while obtaining stability and optimality properties (Mayne *et al.* 2000). Figure 2-7 illustrates the prediction behaviour of an MPC. It shows a reference trajectory to be followed, by calculating the optimal control input, correcting the deviation of this trajectory.

[1]Image downloaded from https://new.abb.com/control-systems/features/model-predictive-control-mpc on January 2019

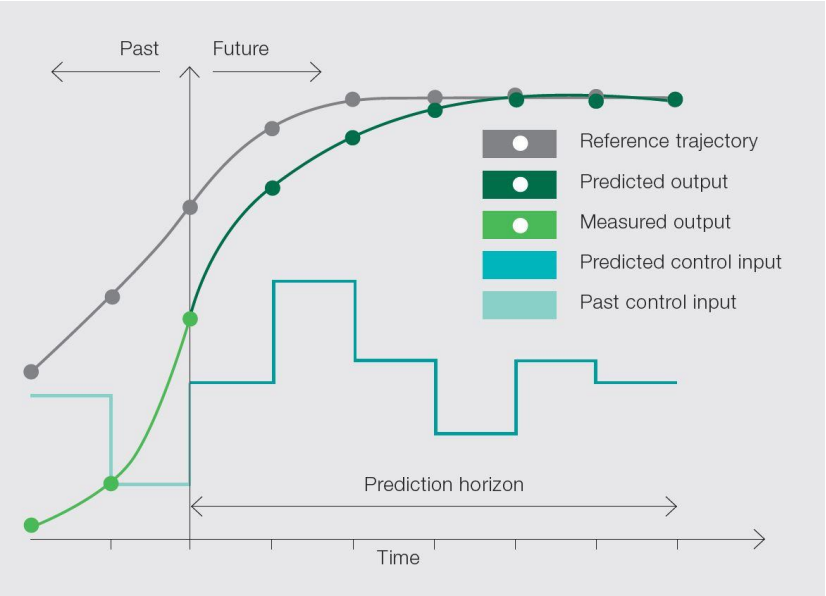


Figure 2-7: An illustrative working of the MPC scheme.[1]

2-3-1 Formulation

Mathematically, the problem is defined in general components. It consists of a cost function, which will be optimised in its arguments, state $\mathbf{x}$ and input $\mathbf{u}$, and constraints in which the problem exists. The prediction horizon at time t will be N steps of length h into the future: $[t, t+Nh]$. The following equations are the basis of an Model Predictive Controller (MPC).

$$\min_{\mathbf{x},\mathbf{u}} J_N(\mathbf{x}_N, \mathbf{u}_N) + \sum_{k=0}^{N-1} J_n(\mathbf{x}_k, \mathbf{u}_k) \tag{2-10}$$

$$\mathbf{x}_0 = \hat{\mathbf{x}}_0 \tag{2-11}$$

$$\mathbf{x}_{k+1} = f(\mathbf{x}_k, \mathbf{u}_k) \tag{2-12}$$

$$h(\mathbf{x}, \mathbf{u}) = 0 \tag{2-13}$$

$$g(\mathbf{x}, \mathbf{u}) \geq 0 \tag{2-14}$$

$$\mathbf{x}_k \in \mathcal{X} \tag{2-15}$$

$$\mathbf{u}_k \in \mathcal{U} \tag{2-16}$$

Equation 2-10 shows an objective function to be minimised, which represents a cost based on the states and input of the system. It consists of two parts, an intermediate and terminal part. The intermediate cost J_n is a sum of all costs of trajectory length $N-1$. The final cost J_N, at time $t+Nh$, is the cost placed at the terminal state of the projected trajectory.

A simple example for the objective function would be a quadratic cost of the states and input, e.g. $J_n = \mathbf{x}^T Q\mathbf{x} + \mathbf{u}^T R\mathbf{u}$. Matrices Q and R are weight matrices and typically diagonal, such that each non-zero entry corresponds to a weight for a specific state. This allows the user to assign more important states or inputs to control. The behaviour of the controller would

drive all states of the system to zero using minimal input. Of course, the user is free to design customised cost functions to create specific behaviour for the controller.

The initial state of this problem changes each iteration. Equation 2-11 relates to the this. It states that initial state $\mathbf{x}_0$ of the current iteration is equal to the measured state $\hat{\mathbf{x}}_0$.

The last mandatory constraint is the use of a dynamical model, as in Equation 2-12. It is used to calculate a trajectory following the system its dynamical model. Moreover, implementing a correct model is very important, as all behaviour originates from this model.

Equations (2-13) to (2-16) will be discussed in the next section.

2-3-2 Constrained control

MPC allows the use of constraints to achieve its goal. Often times, a system is unable to reach a certain state or use every possible input. For example, the system cannot go through walls or obstacles or the allowed input depends on the velocity. These situation can be written as equality or inequality constraints, see Equations 2-13 and 2-14, respectively.

State and input constraints are defined in Equations 2-15 and 2-16. Typically, they construct boundaries for the problem, such as limits of the environment or a maximum amount of current passing through a motor as input.

2-4 Summary

The basis of the inspection to be performed lies within the view global planner. The algorithm by Potdar 2018 calculates a feasible inspection tour, using a mesh object, camera characteristics and a few user defined constraints as input. Using this, only viewable parts of the mesh are included in the tour.

Submodularity is an extension to set functions that describes diminishing returns. Inspection problems can be formulated in a submodular manner, such that greedy heuristics can be applied to find a solution, achieving performance guarantees.

Model predictive control is a powerful tool utilising knowledge of the dynamical model to predict a trajectory. Based on this prediction and the defined cost function, a custom type of behaviour can be achieved.

Table 2-1 summarises the pros and cons of the mentioned methods.

Table 2-1: Summary of the view global planner algorithm and submodularity.

Method	Remarks
View global planner	Requires mesh object
	Includes camera characteristics and user defined constraints.
	Discretisation influences computation time and solution quality.
	Ray casting for precise determination of viewable triangles.
	The generated tour is dynamically feasible and only contains viewable triangles.
Submodularity	Defined by set functions.
	Preservation of submodularity when taking non-negative linear combinations, residuals and truncation.
	Greedy optimisation is the best possible heuristic to guarantee an approximative solution.
	Potentially improves viewpoints generated by the view global planner.
Model predictive control	A custom cost function allows the behaviour of the robot to resemble inspections.
	Constrained control allows for state and input constraints, as well as obstacle avoidance.

Chapter 3

Related works

The research novelty that is addressed within this thesis can be compared with state-of-the-art related to autonomous inspections. In general, three principles can be directed to autonomous inspection: Coverage Path Planning (CPP), covering an area or volume of interest, Informative Path Planning (IPP), seeking to sense a certain amount of information, and Next-best-view (NBV), determining the next feasible viewing location. The last two principles show a lot of overlap, where the difference lies in the exploring nature of NBV.

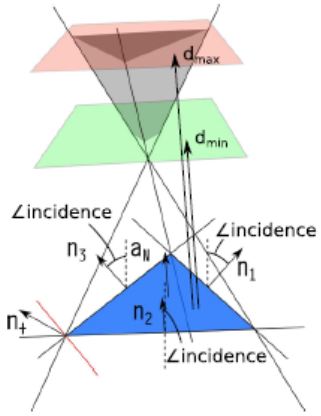
This chapter discusses research papers within these fields and, based on the existing methods, the novelty reached by implementing the solution to the problem stated in Chapter 1. Moreover, the literature review, completed in January 2019, serves as the primary background for this chapter.

3-1 Coverage path planning

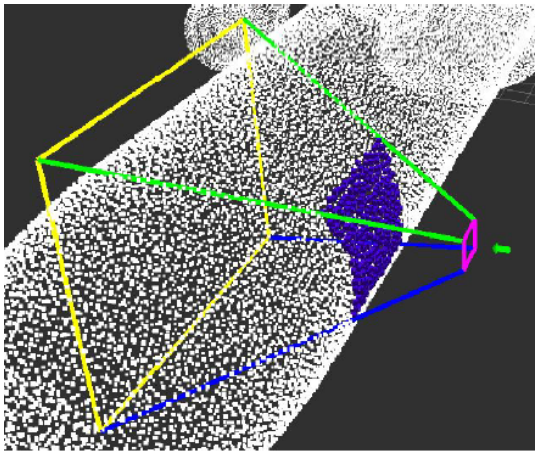
A possible solution to calculate an inspection path covering an object is to generate one viewpoint per triangle and connecting those to obtain a path. Bircher *et al.* 2016c implement this approach by checking if a position is able to sense a triangle (see Figure 3-1(a)), followed by resampling this position to decrease the cost of traversal to the neighbouring viewpoint. Execution of this path is done using a Boundary Value Solver (BVS) for TSP, which incorporates obstacle avoidance in the form of Rapidly-exploring Random Trees (RRT) when a direct connection between viewpoints is not feasible. Moreover, the TSP method classifies it as off-line, and multiple triangles might be visible from a single viewpoint, making it a non-optimal approach.

Another proposed approach is based on the viewing frustum, or Field of View (FOV). By checking if points from a point cloud are contained within the FOV (see Figure 3-1(b)), Almadhoun *et al.* 2016 were able to guarantee a certain coverage of the model and reach resolution completeness, using a Submodular Orienteering Problem (SOP) approach. However, the incidence angle was not considered, since the surface is not oriented due to the use of a point cloud.

A 3D reconstruction can be made of an object by inspection. Using MAVs, Roberts *et al.* 2017 implemented an algorithm collecting as much information within a certain time as possible, based on a SOP approach using camera views in an uniformly sampled grid around a coarse model of the scene. Submodularity was leveraged to obtain a performance guarantee on the set of viewpoints. While this is performed in the outside world, the trajectory was pre-computed off-line (due to Orienteering Problem (OP)) and no on-line obstacle avoidance is used.



(a) The constraints determining a space at which the triangle can be sensed. Each normal vector **n** belongs to a plane that is rotated by the incidence angle. The union of these three planes with the minimum- and maximum distances, d_{min} and d_{max} determine the space. Source: Bircher *et al.* 2016c



(b) The purple dots denote the points of an object located within the FOV. Source: Almadhoun *et al.* 2016

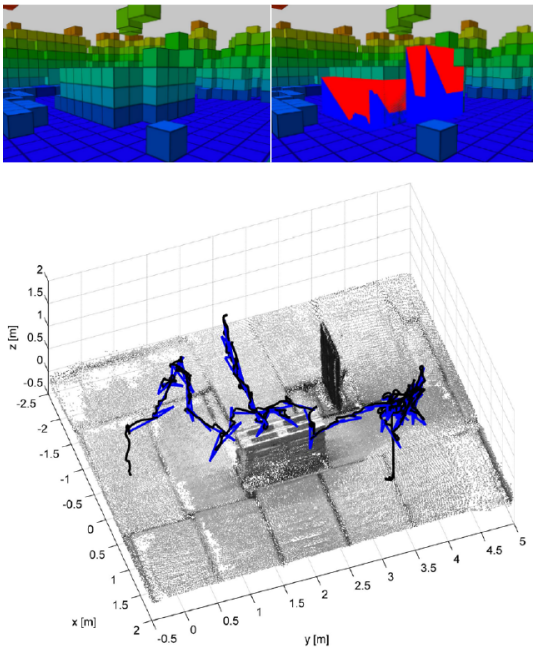
Figure 3-1: Two methods to determine if part of an object can be sensed.

3-2 Informative exploration

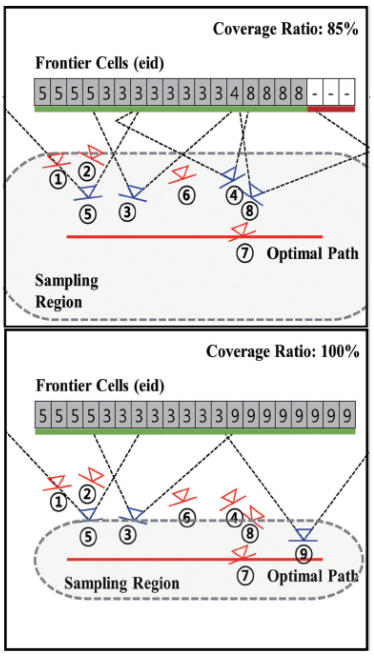
By exploring an unknown environment with a frontier-based approach (regions on the boundary of open- and unexplored space) while avoiding collisions, Bircher *et al.* 2016a and Bircher *et al.* 2016b use an on-line NBV algorithm that plans a path according to sampled views (via RRT) and their information gain. This is a volume based (voxel representation) information gain metric which equals the sum of the unmapped volume or surface that can be explored at the nodes along the branch. Traversal is performed with a trajectory tracking MPC to the first edge towards the view, which results in an obstacle avoiding receding horizon exploration algorithm (see Figure 3-2(a)). However, depending on the resolution of the voxel representation, this approach gives limited information about the occupied voxels, as well as obstacle avoidance. Moreover, the acquired model is based off LIDAR data.

On-line inspection, combined with NBV and voxel representation, is also performed by Lin *et al.* 2017, whose goal is to reduce the unknown set of voxels and generate a desired quality model of the object to inspect. The algorithm optimises a volume-based information gain

by sampling configurations, uses the set cover approach of Emek and Rosén 2014 to find locations that completely inspect the local area and then, finds the shortest route connecting these location by solving TSP. Figure 3-2(b) illustrates the set-cover approach.



(a) In the top left figure, the object to inspect is underneath the voxels in the middle. The top left figure contains red, inspected, parts of the voxels and blue, yet unknown, parts. The lower image shows the resulting inspection path. Source: Bircher *et al.* 2016b



(b) Illustration of the set-cover approach. Samples found within the sampling region that sense new frontier cells are kept as a sub-optimal path. Shrinking the sampling region results and overwriting current samples with better ones results in a more optimal solution. Source: Lin *et al.* 2017

Figure 3-2: Two figures illustrating exploration based methods.

Table 3-1 summarises the pros and cons of the discussed inspection methods.

Table 3-1: Summary of the state-of-the-art algorithms for CPP and informative exploration.

Principle	Literature	Off-/on-line	Mathematical approach	Remarks
CPP	Bircher *et al.* 2016c	Off-line	-	Generates a viewpoint per triangle BVS for path optimisation and obstacle avoidance
	Almadhoun *et al.* 2016	Off-line	SOP	Samples viewpoints in a grid Based on point clouds (no orientation of object) Complete coverage (sub-optimal)
	Roberts *et al.* 2017	Off-line	SOP	3D reconstruction Requires coarse model Considers photogrammetry
NBV	Bircher *et al.* 2016a	On-line	-	Volumetric information gain MPC as waypoint follower Obstacle avoidance by MPC Resolution may limit information gain
	Bircher *et al.* 2016b	On-line	-	Changes information metric to surface information of Bircher *et al.* 2016a gain.
	Lin *et al.* 2017	On-line	Set cover	Samples volumetric information gain for next location Set cover to inspect local area

3-3 Research novelty

The main limitation in the discussed papers, is that the on-line approaches do not obtain some guarantee of performance, due to the exploring nature. On the other hand, the off-line methods do obtain some guarantee, but fall under the problem description in Section 1-3 as observed obstacles are unaccounted for. A real gain here is to combine both an off-line and on-line method to obtain a guarantee. For this reason, the global planner (off-line) and submodularity (on-line) from Chapter 2 are used.

All of the discussed papers use a sampling based approach to determine a point of interest, followed by using a point-to-point controller for movement, when the path is executed. By implementing an MPC, collision avoidance can be achieved. Moreover, carefully designing the cost function enables collisions avoidance and occlusion of obstacles, resulting in a robust inspection.

By designing an information based cost function for the MPC and by leveraging the submodularity property, the MPC drives itself towards new information while achieving performance guarantees. This results in the same or better coverage (when disturbances are encountered) than the pre-computed global path.

The proposed method is on-line, due to MPC, and a novel approach arises that is not sampling-based. In short, the following accounts for the research novelty:

- Combined off-line CPP approach with on-line MPC as inspection method;
- Robust on-line inspection by avoiding collision and occlusion of obstacles;
- Guaranteeing the same or better performance than the pre-computed global path.

Chapter 4

Inspection method

Using the knowledge of the previous chapters, the design of the method can be made. This chapter describes the architecture of the method. In particular, the proposed algorithm will execute an inspection. Here, the aspect of translating position into information is key to the method, where information is the part if the object to be observed. Therefore, attention will be given to the interaction of information between the robot, the object to inspect and obstacles.

Algorithm 1 summarises the proposed method and references corresponding sections, enabling automated inspection in cluttered environments. It iteratively drives the MAV towards to next best viewpoints and takes an image to append the inspected part of the object to inspect. The following objectives are accounted for by the method:

- Use the global planner as a guidance inspection path;
- Include occlusion of obstacles;
- Ensure a feasible plan by implementing collision avoidance.

These objectives are achieved by formulating cost function J, to be fed into an MPC. A cost function is a combination of functions, linear or non-linear, where the states and inputs of the system are parameters, defining the behaviour of the robot. In this case, J accounts for inspection and occlusion.

The method splits the inspection problem into smaller segments, such that each cost function contributes to the final solution and can be solved individually. Furthermore, the method computes real-time trajectories that lead to collision free flight of the MAV.

The derivation of the components of Algorithm 1, such that the custom MPC can be formulated, will be discussed in this chapter. To obtain knowledge of the available states to manipulate, the control architecture will be covered first, followed by the derivation of translating position to information gain. Lastly, the metrics necessary for collision avoidance are derived in Section 4-4.

Algorithm 1 Autonomous inspection

```
1: procedure INSPECTION
2:     [V, M_gp] ← ViewGlobalPlanner(M)                          ▷ Section 2-1
3:     m_infeasible ← ∅
4:     m_feasible ← ∅
5:     for Length(V) do
6:         [m, J] ← Initialise(M_gp, ν_k, O)                     ▷ Section 4-2
7:         while IsFeasible(J) do
8:             FindNextViewpoint(J)                              ▷ Section 4-3
9:             m_new ← TakePhoto()
10:            m_inspected ← Add(m_new)
11:            if Length(m_inspected) = Length(m) then break
12:            else
13:                J ← UpdateCostFunction({m \ m_inspected)}     ▷ Section 4-5
14:        if Length(m_inspected) ≠ Length(m) then
15:            m_infeasible ← Add({m \ m_inspected)})
```

4-1 Control architecture

The control architecture gives insight in the available states and input that must be used to achieve the required behaviour of the robot. Hence, it influences the design of the controller and must be look at first. A schematic block diagram of the control architecture for the MAV is shown in Figure 4-1.

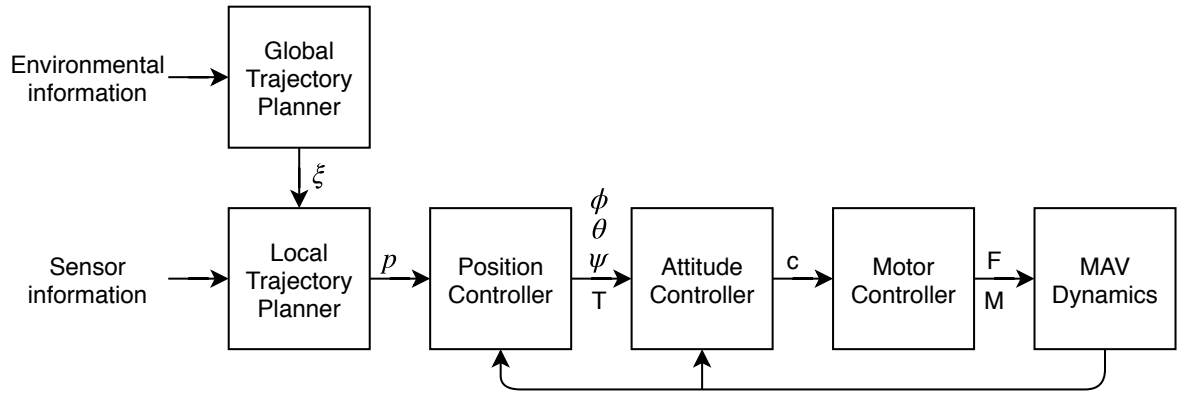


Figure 4-1: A schematic representation of the control architecture.

On the upper level, the global trajectory planner (from Section 2-1) generates path ξ, consisting of the viewpoint set $\mathcal{V}$. This is then fed into a local trajectory planner, which uses the sensor information and generates a goal position. From here, the position controller calculates the required attitudes, the attitude controller tries to reach these and the motor controller translates it into electrical signals for the MAV. Lastly, the dynamics approximate and predict the behaviour.

Low level control The MAV comes with an API referenceDJI 2018, disallowing direct motor control. Talking to the motor can only be done via the attitude controller. Hence, the lowest level signal that can be sent is the attitude of the MAV.

Local controller Since only the attitude can be fed into the MAV, the dynamical model as defined in Equation 1-3, in combination with an MPC, is able to replace the position controller and the local trajectory planner. When merging these two into one, the controller must be designed such that the objective function drives the MAV to the viewpoints generated by the global trajectory planner by solving for input parameters φ, θ, ψ and F_{thrust}.

4-2 Initialising a subproblem

The path ξ received by the global planner can become very large, depending on which object B needs to be inspected. Moreover, the path, observing total area M_{gp} consists only of p viewpoints that contribute a part of M_{gp}. Therefore, it makes sense to split the global inspection problem into p smaller local inspection problems. Figure 4-2 illustrates an example where ξ is split into separate subproblems.

Considering obstacle set $\mathcal{O}$ as disturbances, initialising subproblems depends on current viewpoint ν_k, M_{gp} and $\mathcal{O}$. The output of Line 6 from Algorithm 1 is a copy of subset $m_k \subseteq M$ and cost function J. The mesh observed from viewpoint ν_k corresponds to m and J is a cost function describing information gain and obstacle avoidance. Furthermore, J will be derived in the next sections and formulated in Section 4-5.

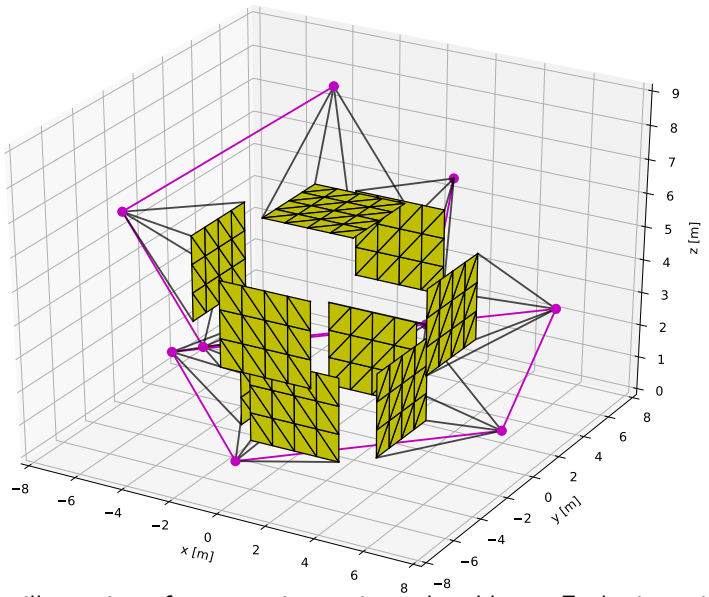
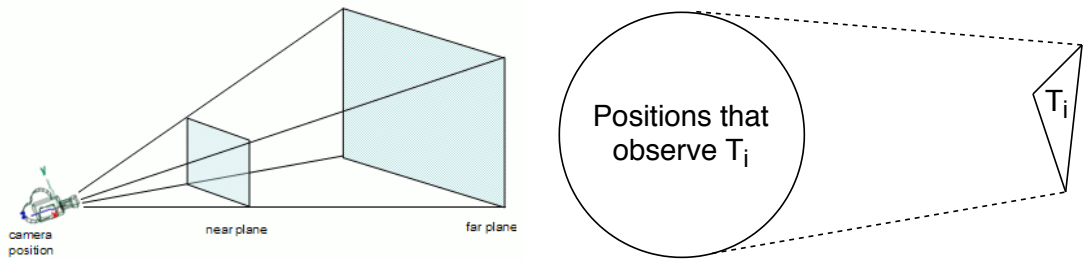


Figure 4-2: An illustrating of separate inspection subproblems. Each viewpoint corresponds to inspecting a certain part of the mesh, shown as an exploded view of the tour.

4-3 Information gain for the next viewpoint

The only available parameters for the cost function are the model- and input states of the system. Therefore, the information gain metric must depend on (a set) those. Appending states to incorporate camera dynamics increases complexity of the problem, which is not desired and is avoided, if possible. Additionally, the MAV must be hovering while taking an image at a viewpoint, meaning that the velocities are unavailable for the cost function. Hence, informativeness must be quantified only with states x, y and z.

Information will be gained when a triangle of the mesh is observable. Two possible methods, briefly discussed in Section 3-1 and illustrated in Figure 4-3, are based on utilising the camera dynamics and an observable space, respectively. The first requires the addition of two states (yaw and pitch of the camera) and is already utilised by the global planner (by the use of ray casting). The second method is to define a confined space in which the triangle is observable, similar to the approach considered by Bircher *et al.* 2016c. This approach avoids appending states and ray casting, reducing the computational complexity, and serves as the basis for the derivation in this section.



(a) The viewing frustum of a camera.[1]

(b) The circle signifies the space of possible positions which observe the triangle T_i.

Figure 4-3: Two figures illustrating methods to deem a triangle viewable.

4-3-1 Defining the observable space

The observable space for a triangle can be defined as the union of eight planar constraints. The parameters required to determine these constraints are the minimum-, maximum distance and maximum incidence angle, d_{min}, d_{max} and δ, respectively. Each distance imposes a single planar constraint and the incidence angle imposes six.

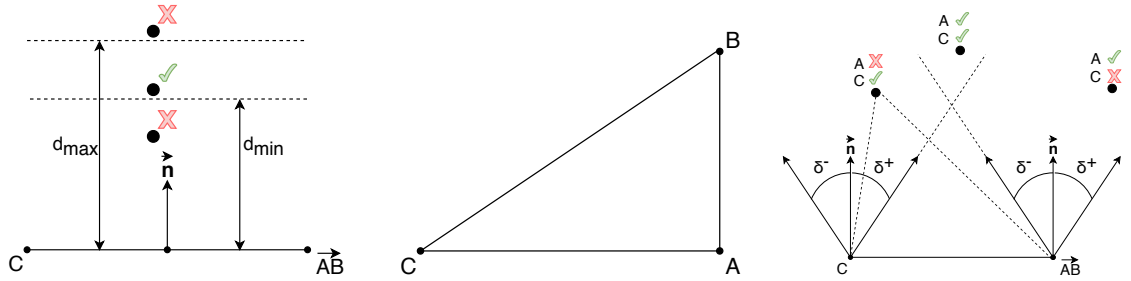
Distance constraints All points on T_i must be within distance d_{min} and d_{max}. For simplicity, the setup illustrated in Figure 4-4(a) is considered. For example, vectors to the constraint planes are $\mathbf{P} - (\mathbf{t} + d_{min}\mathbf{n}_i)$ and $\mathbf{P} - (\mathbf{t} + d_{max}\mathbf{n}_i)$. To determine if $\mathbf{P}$ is before or behind the constraint plane, the dot product with n_i (in the desired direction) must be calculated. The distance falls within the constraints, when the following equations hold:

[1] Image downloaded from http://www.lighthouse3d.com/tutorials/view-frustum-culling/ on February 2019

$$\begin{aligned}(\mathbf{P} - (\mathbf{t} + d_{min}\mathbf{n}_i)) \cdot \mathbf{n}_i > 0 \\ (\mathbf{P} - (\mathbf{t} + d_{max}\mathbf{n}_i)) \cdot -\mathbf{n}_i > 0\end{aligned} \tag{4-1}$$

By writing out the equation and using the length of the normal vector being equal to 1, these constraints can be rewritten to:

$$\begin{aligned}(\mathbf{P} - \mathbf{t}) \cdot \mathbf{n}_i > d_{min} \\ -(\mathbf{P} - \mathbf{t}) \cdot \mathbf{n}_i > -d_{max}\end{aligned} \tag{4-2}$$



(a) Illustrating the minimum- and maximum distance.

(b) An arbitrary triangle with a side view in Figures 4-4(a) and 4-4(c).

(c) Incidence angle normals. Rotating the normals around **AB** will result in planar constraints that tell if all points along the line **C** to **AB** are within the incidence angle. For each dot, the marking indicates whether the angle to points **A** or **C** is smaller or equal than δ

Figure 4-4: Three figures illustrating the imposed constraints to deem a triangle viewable.

Incidence angle constraints The angle between $\mathbf{n}_i$ and $(\mathbf{P} - \mathbf{t})$ must be smaller than δ. Using Figure 4-4(c) as example, it can easily be seen that for a position directly above a point, e.g. **C**, the incidence angle equals 0 regarding that point. However, this is not the case for the opposing edge, **AB**, and can be much larger, as illustrated in Figure 4-4(c). Hence, two constraints must be imposed for each centre line, such that the worst case scenarios fulfil this requirement.

In total, six planar constraints must be derived. The base vectors spanning the constraint planes are the edges and rotated normals of the triangle. This can be achieved by using Rodrigues' rotation formula and the cross product to obtain the normal vectors, formally defined as:

$$\mathbf{v}_{rot} = \mathbf{v}\cos(\theta) + (\mathbf{k} \times \mathbf{v})\sin(\theta) + \mathbf{k}(\mathbf{k} \cdot \mathbf{v})(1 - \cos(\theta)) \tag{4-3}$$

Here, $\mathbf{v}$ is the vector to rotate an angle θ around normalised vector $\mathbf{k}$. Using Figure 4-4(c) as an example, rotating the normal $\mathbf{n}_i$ around vector $\mathbf{AB}$, in both positive- and negative δ direction, yields two rotated vectors that, with $\mathbf{AB}$, span the basis for two planes:

$$\begin{aligned}\mathbf{v}_{i,AB^+} &= \mathbf{n}_i \cos(\delta) + (\mathbf{AB} \times \mathbf{n}_i) \sin(\delta) + \mathbf{AB}(\mathbf{AB} \cdot \mathbf{n}_i)(1 - \cos(\delta)) \\ \mathbf{v}_{i,AB^-} &= \mathbf{n}_i \cos(-\delta) + (\mathbf{AB} \times \mathbf{n}_i) \sin(-\delta) + \mathbf{AB}(\mathbf{AB} \cdot \mathbf{n}_i)(1 - \cos(-\delta))\end{aligned} \tag{4-4}$$

Due to perpendicular vectors for rotation, the dot product in the second half equals zero, resulting in:

$$\begin{aligned}\mathbf{v}_{i,AB^+} &= \mathbf{n}_i \cos(\delta) + (\mathbf{AB} \times \mathbf{n}_i) \sin(\delta) + 0 \\ \mathbf{v}_{i,AB^-} &= \mathbf{n}_i \cos(-\delta) + (\mathbf{AB} \times \mathbf{n}_i) \sin(-\delta) + 0\end{aligned} \tag{4-5}$$

Then, taking the cross product yields normal vectors for the planar constraints of triangle i, applied to points $\mathbf{t}$ and originating from the rotation around $\mathbf{AB}$. The plane rotated in positive direction contains both points $\mathbf{A}$ and $\mathbf{B}$. The plane rotated in negative direction contains point $\mathbf{C}$. For the naming convention, only the first point will be used to define a basis for the planar constraint, i.e. $\mathbf{A}$ and not $\mathbf{B}$:

$$\begin{aligned}\mathbf{n}_{i,C,AB^+} &= -\mathbf{AB} \times \mathbf{v}_{i,AB^+} \\ \mathbf{n}_{i,A,AB^-} &= \mathbf{AB} \times \mathbf{v}_{i,AB^-}\end{aligned} \tag{4-6}$$

Repeating this process for all edges yields six different constraints to determine visibility of T_i at $\mathbf{P}$, based on δ:

$$\begin{aligned}(\mathbf{P} - \mathbf{A}) \cdot \mathbf{n}_{i,A,BC^+} &> 0 \\ (\mathbf{P} - \mathbf{A}) \cdot \mathbf{n}_{i,A,AB^-} &> 0 \\ (\mathbf{P} - \mathbf{B}) \cdot \mathbf{n}_{i,B,CA^+} &> 0 \\ (\mathbf{P} - \mathbf{B}) \cdot \mathbf{n}_{i,B,BC^-} &> 0 \\ (\mathbf{P} - \mathbf{C}) \cdot \mathbf{n}_{i,C,AB^+} &> 0 \\ (\mathbf{P} - \mathbf{C}) \cdot \mathbf{n}_{i,C,CA^-} &> 0\end{aligned} \tag{4-7}$$

4-3-2 Occlusion by obstacles

A technique called horizon culling (*Horizon Culling*) uses simple calculations to determine whether obstacle O is occluding the view of T_i from location $\mathbf{P}$. Two constraints are needed to determine if a single point is occluded or not: whether $\mathbf{P}$ lies in front of- or behind the horizon plane and if $\mathbf{P}$ is located inside or outside of the occlusion cone (see Figure 4-5). For simplicity, $\mathbf{T}_i$ is denoted by its centre and a unit sphere is considered.

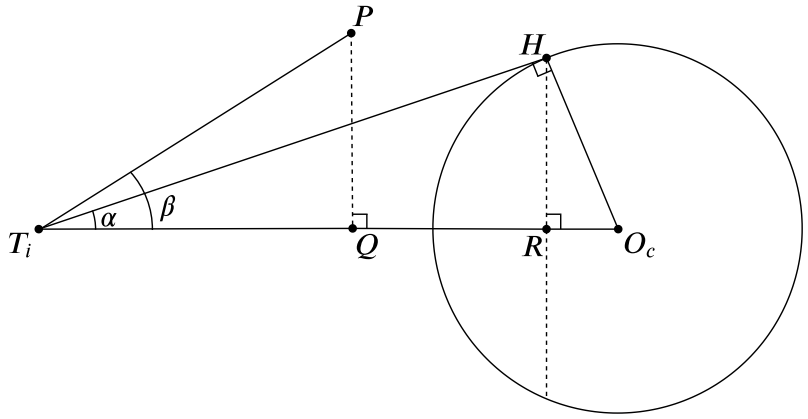


Figure 4-5: Horizon culling for triangle T_i of point $\mathbf{P}$ and obstacle O with centre O_c. $\mathbf{H}$ is the point of the horizon plane. $\mathbf{Q}$ and $\mathbf{R}$ are projections onto $T_i\mathbf{O}_c$ of $\mathbf{P}$ and $\mathbf{H}$, respectively. α and β describe the angle relatives to $T_i\mathbf{O}_c$ of $\mathbf{H}$ and $\mathbf{P}$, respectively. The horizon cone starts in T_i and is directed towards $T_i\mathbf{O}_c$ with angle β.

Assumptions Occlusion by obstacles will be defined as true when the centre of triangle T_i is occluded. This does not cover all other points, e.g. $\mathbf{A}$, $\mathbf{B}$ or $\mathbf{C}$. A simplification has to be made in order to avoid computationally heavy ray casting. However, a non-binary occlusion penalty will be appointed in Chapter 5. Given that this occlusion penalty will be based on the optimal viewing location (straight above the centre of mass), the true optimal viewing location will not be as close to this constraint as possible, loosening the constraint and justifying this assumption.

The horizon plane constraint Figure 4-5 is used as example to derive the constraint. There are two similar triangles, $\triangle\mathbf{T}_i\mathbf{O}_c\mathbf{H}$ and $\triangle\mathbf{O}_c\mathbf{H}\mathbf{R}$, due to the fact they share angle $\angle\mathbf{T}_i\mathbf{O}_c\mathbf{H}$ and both have a right angle, $\mathbf{H}$ and $\mathbf{R}$, respectively. Moreover, the radius of a unit sphere equals 1, implying $\|\mathbf{O}_c\mathbf{H}\| = 1$ and therefore, a relation can be made between $\|\mathbf{R}\mathbf{O}_c\|$ and $\|\mathbf{T}_i\mathbf{O}_c\|$ as follows:

$$\frac{\|\mathbf{R}\mathbf{O}_c\|}{\|\mathbf{O}_c\mathbf{H}\|} = \frac{\|\mathbf{O}_c\mathbf{H}\|}{\|\mathbf{T}_i\mathbf{O}_c\|} \Rightarrow \|\mathbf{R}\mathbf{O}_c\| = \frac{1}{\|\mathbf{T}_i\mathbf{O}_c\|} \tag{4-8}$$

The distance to the horizon plane then becomes:

$$\|\mathbf{T}_i\mathbf{R}\| = \|\mathbf{T}_i\mathbf{O}_c\| - \frac{1}{\|\mathbf{T}_i\mathbf{O}_c\|} \tag{4-9}$$

When the projection of $\mathbf{T}_i\mathbf{P}$ is smaller than $\mathbf{T}_i\mathbf{R}$, the triangle is not occluded by O. Vector projection $\mathbf{T}_i\mathbf{Q}$ is defined as:

$$\mathbf{T}_i\mathbf{Q} = \frac{\mathbf{T}_i\mathbf{P} \cdot \mathbf{T}_i\mathbf{O}_c}{\|\mathbf{T}_i\mathbf{O}_c\|^2}\mathbf{T}_i\mathbf{O}_c \tag{4-10}$$

Then, taking the norm to obtain the length of $\mathbf{T}_i\mathbf{Q}$:

$$\|\mathbf{T}_i\mathbf{Q}\| = \frac{\mathbf{T}_i\mathbf{P} \cdot \mathbf{T}_i\mathbf{O}_c}{\|\mathbf{T}_i\mathbf{O}_c\|^2} \|\mathbf{T}_i\mathbf{O}_c\| \tag{4-11}$$

Arriving at the following inequality for the horizon plane:

$$\frac{\mathbf{T}_i\mathbf{P} \cdot \mathbf{T}_i\mathbf{O}_c}{\|\mathbf{T}_i\mathbf{O}_c\|^2} \|\mathbf{T}_i\mathbf{O}_c\| < \|\mathbf{T}_i\mathbf{O}_c\| - \frac{1}{\|\mathbf{T}_i\mathbf{O}_c\|} \tag{4-12}$$

This can be simplified by multiplying both sides with $\|\mathbf{T}_i\mathbf{O}_c\|$, to obtain the final inequality check:

$$\mathbf{T}_i\mathbf{P} \cdot \mathbf{T}_i\mathbf{O}_c < \|\mathbf{T}_i\mathbf{O}_c\|^2 - 1 \tag{4-13}$$

The occlusion cone constraint When Equation 4-13 turns out to be false, $\mathbf{T}_i$ is occluded from $\mathbf{P}$ when $\alpha > \beta$. Using the dot product to find the cosine of angle β (the sign flips for $0 \leq \phi \leq \pi$) and standard trigonometry, the inequality becomes:

$$\begin{gathered} \cos\beta > \cos\alpha \\ \frac{\mathbf{T}_i\mathbf{P} \cdot \mathbf{T}_i\mathbf{O}_c}{\|\mathbf{T}_i\mathbf{P}\| \, \|\mathbf{T}_i\mathbf{O}_c\|} > \frac{\|\mathbf{T}_i\mathbf{H}\|}{\|\mathbf{T}_i\mathbf{O}_c\|} \end{gathered} \tag{4-14}$$

Which simplifies to:

$$\frac{\mathbf{T}_i\mathbf{P} \cdot \mathbf{T}_i\mathbf{O}_c}{\|\mathbf{T}_i\mathbf{P}\|} > \|\mathbf{T}_i\mathbf{H}\| \tag{4-15}$$

An expression for $\|\mathbf{T}_i\mathbf{H}\|$ can be found using Pythagoras:

$$\|\mathbf{T}_i\mathbf{H}\|^2 = \|\mathbf{T}_i\mathbf{O}_c\|^2 - 1 \tag{4-16}$$

Finally, squaring Equation 4-15 and substituting Equation 4-16, results in the occlusion cone constraint:

$$\frac{(\mathbf{T}_i\mathbf{P} \cdot \mathbf{T}_i\mathbf{O}_c)^2}{\|\mathbf{T}_i\mathbf{P}\|^2} > \|\mathbf{T}_i\mathbf{O}_c\|^2 - 1 \tag{4-17}$$

Summarising, $\mathbf{T}_i$ is occluded by O when Equation 4-13 is false and Equation 4-17 is true.

Generalising to ellipsoids A simple extension can be made towards the ability of using the same occlusion constraints with an ellipsoid, which general equation was defined in Equation 1-4. A space relative to the ellipsoid can be made by dividing a vector by the radii $\mathbf{r}_O$ of O, such that the ellipsoid is represented as a unit sphere and all vectors are relative.

4-3-3 Scoring the information gain

Being able to just observe a triangle is binary, either 1 or 0, based on the previous sections. Triangles can differ in size, hence it makes sense to apply the area $A(T_i)$ as a weight to the cost. Moreover, the incidence angle γ is also a good indicator. Being straight above T_i must give a higher score than sideways. Hence, the cosine fits the description because it is normalised (assuming $-\pi/2 \leq \gamma \leq \pi/2$) and defined as:

$$\cos\gamma = \frac{\mathbf{T}_i\mathbf{P} \cdot \mathbf{n}_i}{\|\mathbf{T}_i\mathbf{P}\|} \tag{4-18}$$

Being closer to T_i results in a higher score. Normalising this distance with respect to d_{min} and d_{max} results in the following distance score:

$$d_{score} = \frac{d_{max} - \mathbf{T}_i\mathbf{P} \cdot \mathbf{n}_i}{d_{max} - d_{min}} \tag{4-19}$$

Finally, the score for each triangle is:

$$s_i(\mathbf{P}) = \begin{cases} A(T_i)\left(w_{incidence}\frac{\mathbf{T}_i\mathbf{P}\cdot\mathbf{n}_i}{\|\mathbf{T}_i\mathbf{P}\|} + w_{height}\frac{d_{max}-\mathbf{T}_i\mathbf{P}\cdot\mathbf{n}_i}{d_{max}-d_{min}}\right), & \text{if } T_i \text{ is observable from } \mathbf{P} \\ 0, & \text{otherwise} \end{cases} \tag{4-20}$$

Where $w_{incidence}$ and w_{height} are defined as weights with respect to the incidence angle and height, respectively. The total information gain score for viewing a triangle then becomes:

$$I(\mathbf{P}) = \sum_{i=0}^{n} s_i(P) \tag{4-21}$$

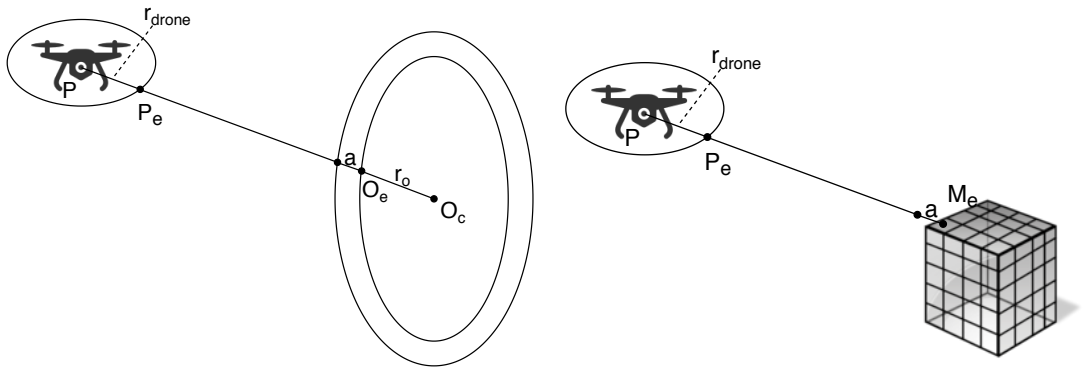
Here, n equals the amount of triangles. Clearly, maximising the information gain is the desired task.

4-4 Collision avoidance

Two things must be avoided in the environment of the MAV, obstacles and the object to inspect. Therefore, two distance metrics are derived that penalise coming too close. Figures 4-6(a) and 4-6(b) illustrate how these metrics are defined. For customisability, the user can define an activation distance a (in meters), which is the range in which the penalty becomes active.

4-4-1 Obstacle distance

The MAV is represented as a bounding sphere and a penalty must be generated when it enters a certain region. The distance between bounding sphere of the MAV and ellipsoid of



(a) The distance between two ellipsoids. $\mathbf{P}$ and $\mathbf{O}_c$ are the centres, $\mathbf{P}_e$ and $\mathbf{O}_e$ the edges and r_{drone} and $\mathbf{r}_O$ the radii of the ellipsoids containing the MAV and obstacle, respectively. a is the user defined activation distance.

(b) The distance between MAV and mesh. $\mathbf{P}$ is the centre, $\mathbf{P}_e$ the edge and r_{drone} the radii of the ellipsoid containing the MAV. $\mathbf{M}_e$ is the centre of a triangle and a the user defined activation distance.

Figure 4-6: Two figures illustrating generalised distances to obstacles and the mesh.

the obstacle must be derived. Using ellipsoids, as illustrated in Figure 4-6(a), a conversion to the relative space of the obstacle must be performed.

First, the relative vector from the MAV to the centre of the obstacle, $\mathbf{PO}'_c$, needs to be determined. This can be done by dividing $\mathbf{PO}_c$ with the radii of O:

$$\mathbf{PO}'_c = \frac{\mathbf{O}_c - \mathbf{P}}{\mathbf{r}_O} \tag{4-22}$$

Then, multiplying $\mathbf{PO}'_c$ with a scaling factor of the vector and returning to the regular space allows to obtain the location of the edge of the bounding ellipsoid $\mathbf{PO}'_e$:

$$\mathbf{PO}'_e = \frac{\|\mathbf{PO}'_c\| - 1}{\|\mathbf{PO}'_c\|}\mathbf{PO}'_c \tag{4-23}$$

To go back to the original space, $\mathbf{PO}'_e$ is multiplied with $\mathbf{r}_O$ to obtain the distance to O:

$$d_{obst} = \mathbf{PO}'_e\mathbf{r}_O \tag{4-24}$$

Finally, a repulsive force is applied as potential fields (Khatib 1985):

$$d_O(\mathbf{P}) = \begin{cases} \frac{1}{2}(\frac{1}{d_{obst}} - \frac{1}{a+r_{\text{drone}}})^2 & d_{obst} \leq a + r_{\text{drone}} \\ 0 & \text{else} \end{cases} \tag{4-25}$$

4-4-2 Mesh distance

The distance to the mesh is straightforward. It is described as the distance from $\mathbf{P}$ to the centre of triangle $\mathbf{T}_i$:

$$d_{T,i} = \|\mathbf{P}\mathbf{T}_i\| \tag{4-26}$$

Again, a repulsive force is applied by using potential fields to obtain the distance penalty:

$$d_{M,i}(\mathbf{P}) = \begin{cases} \frac{1}{2}(\frac{1}{d_{T,i}} - \frac{1}{a+r_{\text{drone}}})^2 & d_{T,i} \leq a + r_{\text{drone}} \\ 0 & \text{else} \end{cases} \tag{4-27}$$

Here, $a + r_{drone}$ represents the minimal distance at which the repulsive for of the potential field comes into play. The total penalty will then be the sum of distance penalties over all triangles:

$$d_M(\mathbf{P}) = \sum_{i=0}^{N} d_{M,i}(\mathbf{P}) \tag{4-28}$$

4-5 MPC formulation

For each iteration in Algorithm 1, a cost function must be set. It is comprised of three terms and defines the desired behaviour of inspecting a mesh without colliding. Objective function J is defined as the information gain and collision penalties of both the obstacles and mesh, as defined in Equation 4-21, 4-25 and 4-28, respectively:

$$J(\mathbf{x}_k, \mathbf{u}_k) = w_I I(\mathbf{x}_k) + w_O d_O(\mathbf{x}_k) + w_M d_M(\mathbf{x}_k) \tag{4-29}$$

Here, $w_I, w_O, w_M > 0$ are customisable weights to be set by the user, each influencing the importance of the corresponding term. Note here, that the information gain is a function to be maximised. Translating this term into a suitable optimisation problem for minimisation, therefore MPC, will be treated in Chapter 5.

The custom formulation of the MPC, for the scope of this thesis, becomes the following:

$$\min_{\mathbf{x},\mathbf{u}} \sum_{k=0}^{N} J(\mathbf{x}_k, \mathbf{u}_k) \tag{4-30}$$

$$\begin{aligned}
\text{subject to} \quad & \mathbf{x}_0 = \hat{\mathbf{x}}_0 && \text{(initial state)} && \text{(4-31)} \\
& \mathbf{x}_{k+1} = f(\mathbf{x}_k, \mathbf{u}_k) && \text{(dynamics)} && \text{(4-32)} \\
& \mathbf{x}_k \in \mathcal{X} && \text{(state constraints)} && \text{(4-33)} \\
& \mathbf{u}_k \in \mathcal{U} && \text{(input constraints)} && \text{(4-34)}
\end{aligned}$$

Here, Equation 4-31 denotes the measured state, which will be set as the initial state at the start of the iteration, Equation 4-32 are the dynamics stated in Equation 1-3, Equations 4-33 and 4-34 denote bounding constraints regarding the state- and input vectors following from Equations 1-1 and 1-2 and Table 1-3, respectively.

Note that the MPC is formulated without hard constraints for collision avoidance. By penalising the terms in the cost function sufficiently, it is assumed the calculated trajectories do not cause collision.

The final method, also described in Algorithm 1, is then achieved by following these iterative steps:

1. The MPC will drive the MAV to the solution of J, acting as a viewpoint.
2. Taking an image at this location adds a node to the set of viewpoints. The new node corresponds to a non-inspected subset of the mesh.
3. Following Section 2-2, the remaining non-inspected mesh can then be greedily maximised again, going back to step 1.

For an inspection, this results in a recursive submodular procedure. According to Krause and Golovin 2014 (Section 1.2), submodularity is preserved when taking non-negative linear combinations of submodular functions. Using this philosophy, looping through each inspection subproblem is the addition of submodular problems and results in a submodular global inspection problem.

Chapter 5

Optimising the information gain

The current information gain $I(\mathbf{P})$ is very hard to optimise, due to planar constraints causing jumps in function evaluations, discarding the continuity and the ability to use gradient-based optimisation algorithms. This chapter extends the defined information gain from Chapter 4 and rewrites it into an optimisable function, suitable with the MPC.

In order to re-introduce differentiability, sigmoid functions are used to approximate the constraints and cast the information gain in a optimisable minimisation problem. A sigmoid function[1], often used in neural networks as activation functions, is a function whose output has an 'S' shape and lies between 0 and 1. Correctly placing such a function on the location of the constraint allows approximation of constraints, while keeping differentiability. Equation 5-1 shows the definition and Figure 5-1 the corresponding curve.

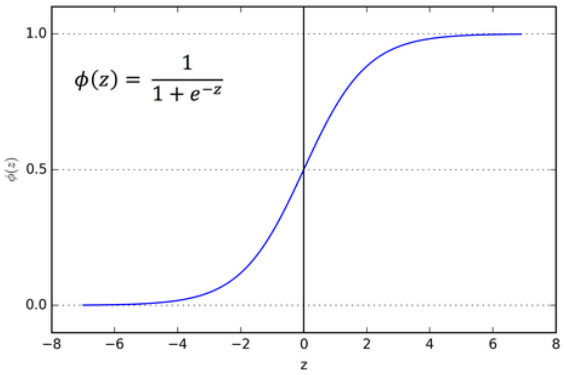


Figure 5-1: The sigmoid curve[1], corresponding to Equation 5-1 with slope $\eta = 1$.

$$s(x) = \frac{1}{1 + e^{-\eta x}} \tag{5-1}$$

Where η is the slope of the sigmoid function and when set to a very high constant, the output resembles a step function.

Sigmoid functions can also be used change the information maximisation- to a minimisation problem, as is discussed in Section 5-1. Furthermore, every constraint derived in Chapter 4 will be converted to a continuous sigmoid function, with the added bonus of implicit optimisation of the incidence angle.

[1] Downloaded from https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6 on June 2019

5-1 Converting maximisation to minimisation

In order to be able to use the information metric in the MPC framework, the maximisation of the submodular information gain function must be transformed to a minimisation problem. Sigmoid functions lend themselves greatly for this purpose and a maximisation problem can be rewritten to minimisation as follows. By using the following symmetry:

$$\begin{aligned} f(x) &= 1 - f(-x) \\ \frac{1}{1+e^{-x}} &= 1 - \frac{1}{1+e^{x}} \\ &= \frac{e^{x}}{1+e^{x}} \\ &= \frac{1}{1+e^{-x}} \end{aligned} \tag{5-2}$$

the maximisation problem can easily be changed to a minimisation problem:

$$\begin{aligned} \max_{x} f(x) &= \min_{x} -f(x) \\ &= \min_{x} -1 + f(-x) \\ &= \min_{x} f(-x) \end{aligned} \tag{5-3}$$

Since $I(\mathbf{P})$ is a sum of sigmoid functions, only the direction of x needs to be changed to make it suitable for the MPC.

5-2 Projecting the constraints

A sigmoid function is one-dimensional. Therefore, the position needs to be projected onto a one-dimensional line. This translates to the minimal distance from $\mathbf{P}$ to a constraint plane. Visualisations for the eight constraints defined in Section 4-3 can be found in Figures 5-2 and 5-3.

5-2-1 Height constraints

According to Figure 5-2, the height with respect to the minimum- and maximum distance needs to be derived. Similar to Equation 4-2, the distance d to T_i is formulated by the following equation:

$$d(\mathbf{P}) = \frac{(\mathbf{P} - \mathbf{t}) \cdot \mathbf{n}_i}{\|\mathbf{n}_i\|^2} = (\mathbf{P} - \mathbf{t}) \cdot \mathbf{n}_i \tag{5-4}$$

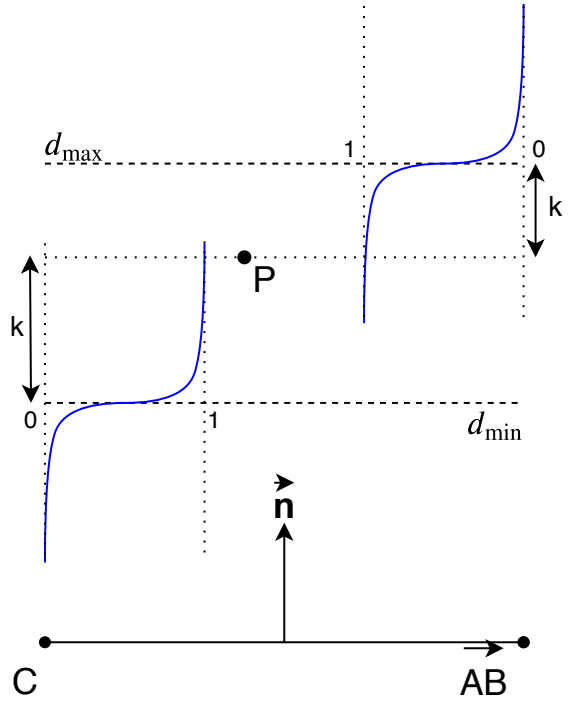


Figure 5-2: Projecting the height constraint, using Figure 4-4(a) as example.

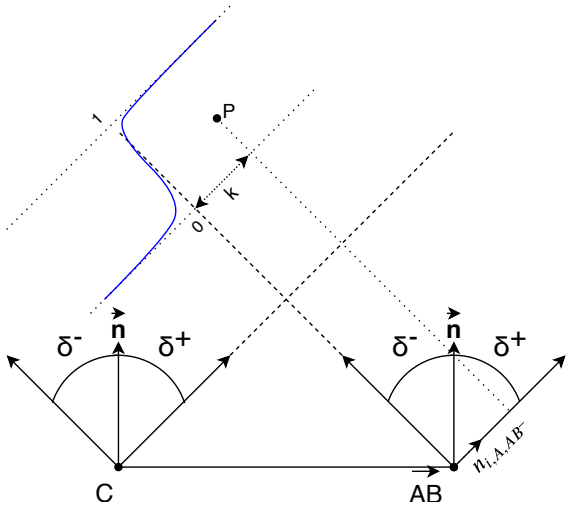


Figure 5-3: Projecting the incidence angle constraint, using Figure 4-4(c) as example.

The distance to the constraint plane is located exactly d_{min} above T_i. Therefore, any point $\mathbf{t} + d_{min}\mathbf{n}_i$ lies on the constraint plane and the distance to the plane becomes:

$$k = \mathbf{P} \cdot \mathbf{n}_i - (\mathbf{t} + d_{min}\mathbf{n}_i) \cdot \mathbf{n}_i \tag{5-5}$$

Here, $\mathbf{P} \cdot \mathbf{n}_i$ represents the distance with respect to the position and $(\mathbf{t} + d_{min}\mathbf{n}_i) \cdot \mathbf{n}_i$ the distance to the plane. $(\mathbf{t} + d_{min}\mathbf{n}_i)$ will be called the hook $\mathbf{h}_c$ of the constraint. The sigmoid function for the minimal distance is defined as:

$$s_{constr}(\mathbf{P}) = \frac{1}{1 + e^{\eta((\mathbf{P} - (\mathbf{t} + d_{min}\mathbf{n}_i)) \cdot \mathbf{n}_i)}} \tag{5-6}$$

Where η is the slope of the sigmoid function. A general formulation for approximating planar constraints is:

$$s(\mathbf{P}, \mathbf{h}_c, \mathbf{n}_i) = \frac{1}{1 + e^{\eta((\mathbf{P} - \mathbf{h}_c) \cdot \mathbf{n}_i)}} \tag{5-7}$$

To obtain the sigmoid function for the maximum distance constraint, simply replace d_{min} with d_{max}.

5-2-2 Incidence angle constraints

The general sigmoid function described in Equation 5-7 can easily be extrapolated to the incidence angle constraints. Recall that, using Figure 5-3, $\mathbf{n}_{i,A,AB^-}$ is the normal vector

corresponding to the plane created by **AB** combined with the resulting vector of rotating $\mathbf{n}_i$ around **AB**, by a negative δ angle, and originating from **A**. Since the constraint plane originates from **A**, this will be the hook of the constraint. The normal constraint logically becomes the corresponding vector $\mathbf{n}_{i,A,AB^-}$.

5-3 Implicit optimisation of sigmoid slopes

Using sigmoid functions also introduces the possibility to implicitly optimise the height and incidence angle by manipulating slope η of the sigmoid function. The optimal location for observability, regarding the height and incidence angle, is as close to T_i as possible and straight above the centre of mass location, respectively. This section will derive the implicit optimisation of the height and incidence angle.

5-3-1 Deriving the optimal location

A simple example is used to derive η for an optimal location. Figure 5-4 shows two sigmoid functions acting as constraints, one shifted to the left by 1 in the positive x-direction and the other reversed and shifted to the right by 1 in the negative x-direction. The total score equals both functions summed up and the ideal location is located exactly in the middle.

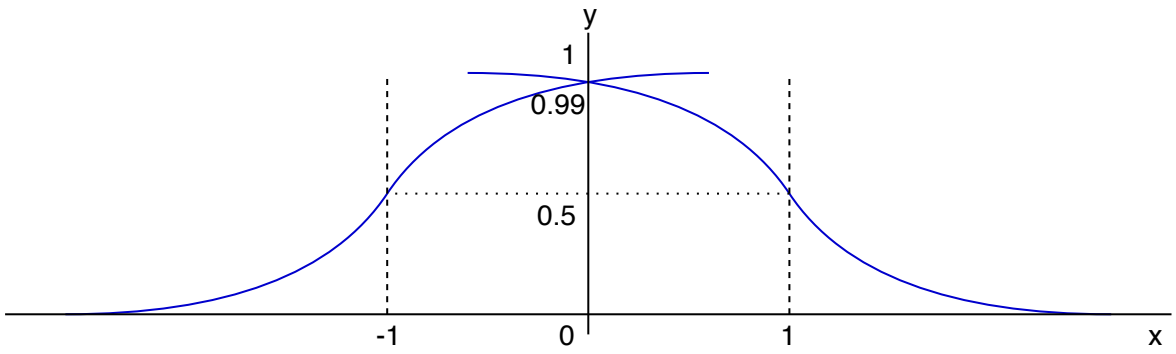


Figure 5-4: Two sigmoid functions, $s_1(x)$ (left) and $s_2(x)$ (right) in opposite direction with different slope η. Implicitly, the slope gives an optimal location.

For this example, the origin should be the optimal location, or 0.99 (constant c). By solving $s_1(0) = 0.99$, η can be found:

$$\begin{aligned} s_1(0) = \frac{1}{1+e^{\eta-(0+1)}} &= 0.99 \\ 1 &= 0.99(1+e^{\eta-(0+1)}) \\ 0.01 &= 0.99e^{-\eta} \\ \frac{1}{99} &= e^{-\eta} \\ \ln(99) &= \eta \end{aligned} \tag{5-8}$$

Since the sigmoid functions in this example are symmetrical, both slopes are the same. However, a trade-off has to be made on choosing constant c. First, it can not be 0.5 and 1,

because those belong to positive and negative infinity. The slope can be chosen arbitrarily as $0.5 < c < 1$. But without the risk of flattening the function, 0.99 seems to be a reasonable standard.

5-3-2 Generalising to height constraints

The optimal location would be as close to the triangle as possible. Therefore, the slopes corresponding to the minimum- and maximum distance should adhere to this. A tuning parameter d_{opt} is introduced, relating to the percentage of allowable distance where the optimal location must be. For instance, choosing $d_{opt} = 0.05$ and having allowable heights from 1 to 6 metres means the optimal location will be at $(6-1)(1-0.05)+1 = 1.25m$. The optimal distances with regard to the sigmoid functions become:

$$x_{opt,min} = (d_{max} - d_{min})d_{opt} \tag{5-9}$$

and

$$x_{opt,max} = (d_{max} - d_{min})(1 - d_{opt}) \tag{5-10}$$

Where solving $s(x_{opt,min}) = c$ and $s(x_{opt,max}) = c$ (as per Section 5-3-1) will yield the corresponding slopes:

$$\eta_{d,min} = \frac{\ln(99)}{(d_{max} - d_{min})d_{opt}} \tag{5-11}$$

and

$$\eta_{d,max} = \frac{\ln(99)}{(d_{max} - d_{min})(1 - d_{opt})} \tag{5-12}$$

Following the same procedure, the sigmoid functions become:

$$s_{d,min}(\mathbf{P}, \mathbf{h}_c, \mathbf{n}_i) = \frac{1}{1+e^{-\frac{\ln(99)}{(d_{max}-d_{min})d_{opt}}(\mathbf{P}-\mathbf{h}_C)\cdot\mathbf{n}_i}} \tag{5-13}$$

and

$$s_{d,max}(\mathbf{P}, \mathbf{h}_c, \mathbf{n}_i) = \frac{1}{1+e^{-\frac{\ln(99)}{(d_{max}-d_{min})(1-d_{opt})}(\mathbf{P}-\mathbf{h}_C)\cdot\mathbf{n}_i}} \tag{5-14}$$

5-3-3 Generalising to incidence angle constraints

Using Figure 5-5, the generalisation to incidence angles can be derived. For any position, the height changes η. Being higher results in a lower slope coefficient if the optimal location is above the centre of mass, which lies on s, or $2/3L$ of the perpendicular vector from $\mathbf{C}$ to edge $\mathbf{AB}$. η must be derived such that evaluating $s(\mathbf{P})$ results in a real value reflecting the deviation of being above s (evaluating results in $c = 0.99$ when directly above).

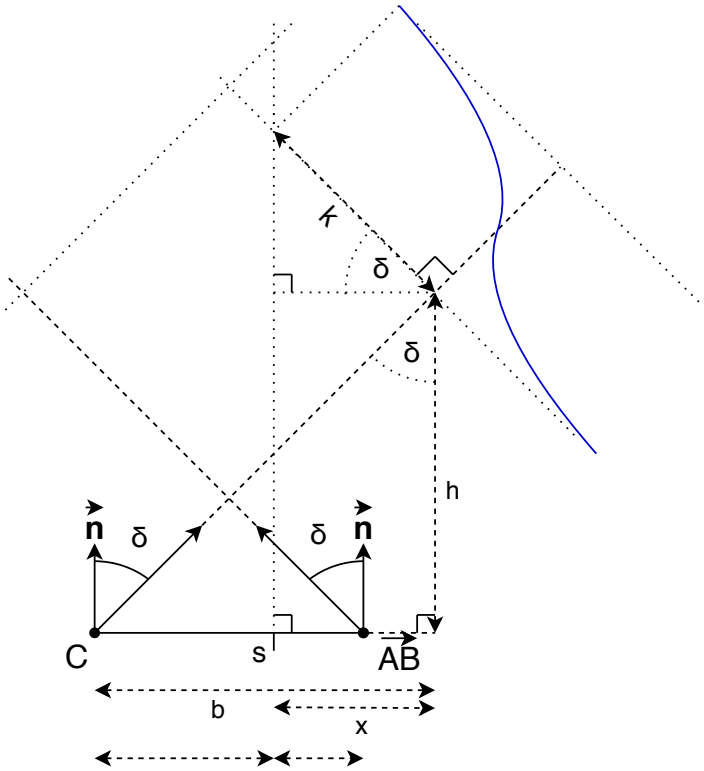


Figure 5-5: Projecting the sigmoid function on the constraint to determine the slope, variable per height.

Length k can be found using x and δ:

$$k = \frac{x}{\cos(\delta)} \tag{5-15}$$

x, b and h can be expressed as:

$$x = b - \frac{2}{3}L \tag{5-16}$$

$$b = h \tan \delta \tag{5-17}$$

$$h = (\mathbf{P} - \mathbf{A}) \cdot \mathbf{n}_i \tag{5-18}$$

Now, substituting Equations 5-16 to 5-18 into Equation 5-15:

$$k = \frac{(\mathbf{P} - \mathbf{A}) \cdot \mathbf{n}_i \tan(\delta) - \frac{2}{3}L}{\cos(\delta)} \tag{5-19}$$

Solving a sigmoid function using k, like Section 5-3-1, entails the following slope η:

$$\eta = \frac{\ln 99 \cos(\delta)}{(\mathbf{P} - \mathbf{A}) \cdot \mathbf{n}_i \tan(\delta) - \frac{2}{3}L} \tag{5-20}$$

This shows η is dependent on the triangle its size and the current position, as expected. Regarding optimisation, $\ln(99)$ and δ are constants throughout the complete inspection, and therefore only need to be calculated once, ridding the problem of expensive tan and cos calculations. The final equation for planar constraints approximated by sigmoid functions become:

$$s(P) = \frac{1}{1+e^{-\frac{\ln 99 \cos(\delta)}{(\mathbf{P}-\mathbf{A})\cdot\mathbf{n}_i \tan(\delta)-\frac{2}{3}L}(\mathbf{P}-\mathbf{A})\cdot\mathbf{n}_{i,C,AB+}}} \tag{5-21}$$

Finally, the sigmoid function approximation originating from A is defined as:

$$s(P) = \frac{1}{1+e^{-\frac{\ln(99) \cos(\delta)}{(\mathbf{P}-\mathbf{A})\cdot\mathbf{n}_i \tan(\delta)-\frac{1}{3}L}(\mathbf{P}-\mathbf{A})\cdot\mathbf{n}_{i,A,AB-}}} \tag{5-22}$$

5-4 Extending to obstacles

Obstacle constraints take action from the horizon plane and occlusion cone. The former is a linear constraint, whereas the latter is non-linear. Using standard geometry, the distance to the plane and cone can be determined without the use of trigonometric functions. Then, by multiplying the two sigmoid functions that follow from the constraints, the occlusion space is obtained. Figure 5-6 shows a set-up, which is used for the derivation.

5-4-1 Horizon plane

Following the analogy of Section 5-2, the required parameters are the hook- and normal of the constraint. The latter is straightforward, namely the normalised vector of $\mathbf{T}_i\mathbf{O}_c$:

$$\mathbf{n}_{\text{horizon}} = \frac{\mathbf{T}_i\mathbf{O}_c}{\|\mathbf{T}_i\mathbf{O}_c\|} \tag{5-23}$$

Note that the direction is the same as $\mathbf{T}_i\mathbf{O}_c$, because the triangle is occluded only when $\mathbf{P}$ is behind the horizon plane.

The hook can be defined as $\mathbf{h}_{\text{horizon}} = \mathbf{O}_c - \mathbf{O}_c\mathbf{R}$. $\mathbf{O}_c\mathbf{R}$ and is derived using Equation 4-8:

$$\mathbf{O}_c\mathbf{R} = \frac{\|\mathbf{R}\mathbf{O}_c\|}{\|\mathbf{T}_i\mathbf{O}_c\|}\mathbf{T}_i\mathbf{O}_c = \frac{\mathbf{T}_i\mathbf{O}_c}{\|\mathbf{T}_i\mathbf{O}_c\|^2} \tag{5-24}$$

Resulting in:

$$\mathbf{h}_{\text{horizon}} = \mathbf{O}_c - \frac{\mathbf{T}_i\mathbf{O}_c}{\|\mathbf{T}_i\mathbf{O}_c\|^2} \tag{5-25}$$

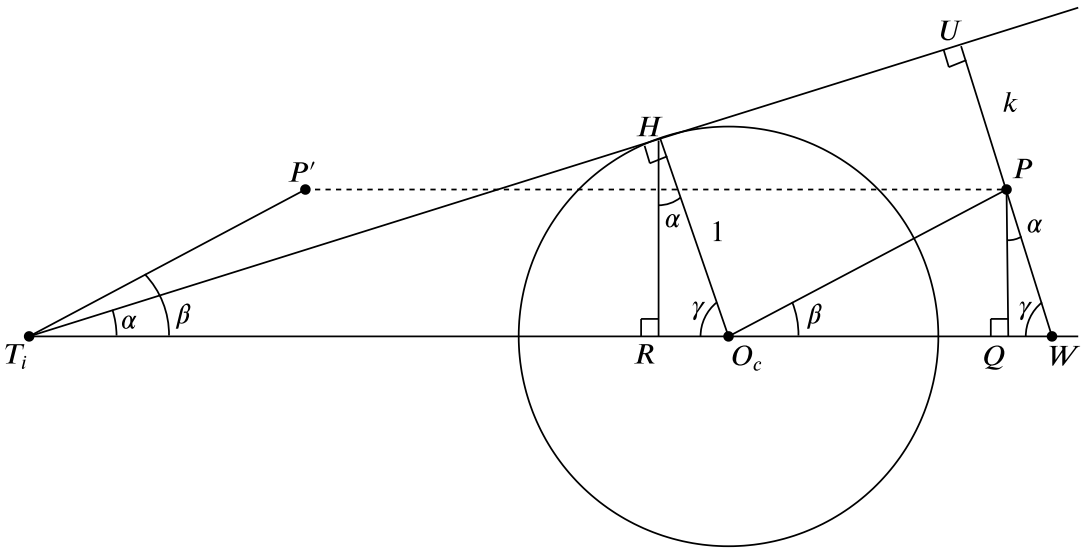


Figure 5-6: Geometry for sigmoid projection of obstacle occlusion. This image is an extension to Figure 4-5, where $\mathbf{UW}$ is perpendicular to the horizon cone through $\mathbf{P}$, and $\mathbf{W}$ is the intersection to the extension of $\mathbf{T}_i\mathbf{O}_c$. $\mathbf{S}$ is the relative projection of $\mathbf{UW}$ onto $\mathbf{HO}_c$.

5-4-2 Occlusion cone

The cone constraint is quite different due to being non-linear. The distance k needs to be derived, which is defined as the shortest distance to the horizon cone. A set of similar triangles can be found in Figure 5-6, namely $\triangle\mathbf{T}_i\mathbf{UW} = f \triangle \mathbf{T}_i\mathbf{HO}_c$. Therefore, the following relation arises:

$$\|\mathbf{UW}\| = f \|\mathbf{HO}_c\| = f \tag{5-26}$$

Here, f is defined by the factor between $\mathbf{T}_i\mathbf{O}_c$ and $\mathbf{T}_i\mathbf{W}$:

$$f = \frac{\|\mathbf{T}_i\mathbf{W}\|}{\|\mathbf{T}_i\mathbf{O}_c\|} \tag{5-27}$$

Which can be rewritten to:

$$f = \frac{\|\mathbf{T}_i\mathbf{O}_c\| + \cos(\beta) \|\mathbf{O}_c\mathbf{P}\| + \cos(\gamma) \|\mathbf{PW}\|}{\|\mathbf{T}_i\mathbf{O}_c\|} \tag{5-28}$$

Next, $\cos(\beta)$ is found via the dot product and $\cos(\gamma)$ follows from standard geometry and Equation 4-8:

$$\cos(\beta) = \frac{\mathbf{O}_c\mathbf{P} \cdot \mathbf{T}_i\mathbf{O}_c}{\|\mathbf{O}_c\mathbf{P}\| \|\mathbf{T}_i\mathbf{O}_c\|} \tag{5-29}$$

$$\cos(\gamma) = \|\mathbf{RO}_c\| = \frac{1}{\|\mathbf{T}_i\mathbf{O}_c\|} \tag{5-30}$$

Substituting into 5-28 and simplifying results in the following relation for f:

$$f = 1 + \frac{\mathbf{O}_c\mathbf{P} \cdot \mathbf{T}_i\mathbf{O}_c + \|\mathbf{PW}\|}{\|\mathbf{T}_i\mathbf{O}_c\|^2} \tag{5-31}$$

By using the sine rule, an expression for $\|\mathbf{PW}\|$ is obtained:

$$\|\mathbf{PW}\| = \frac{\sin(\beta)}{\sin(\gamma)} \|\mathbf{O}_c\mathbf{P}\| \tag{5-32}$$

Now, squaring the above equation combined with using $\sin^2(a) = 1 - \cos^2(a)$ and Equations 5-29 and 5-30, the equation above can be rewritten to:

$$\|\mathbf{PW}\| = \sqrt{\frac{1 - \left(\frac{\mathbf{O}_c\mathbf{P} \cdot \mathbf{T}_i\mathbf{O}_c}{\|\mathbf{O}_c\mathbf{P}\|\|\mathbf{T}_i\mathbf{O}_c\|}\right)^2}{1 - \frac{1}{\|\mathbf{T}_i\mathbf{O}_c\|^2}} \|\mathbf{O}_c\mathbf{P}\|^2} \tag{5-33}$$

Finally, k_{cone} equals:

$$k_{cone} = \|\mathbf{UW}\| - \|\mathbf{PW}\| = f - \|\mathbf{PW}\| \tag{5-34}$$

This, when using Equations 5-31, can be rewritten to:

$$\begin{aligned} k_{cone} &= 1 + \frac{\mathbf{O}_c\mathbf{P} \cdot \mathbf{T}_i\mathbf{O}_c + \|\mathbf{PW}\|}{\|\mathbf{T}_i\mathbf{O}_c\|^2} - \|\mathbf{PW}\| \\ &= 1 + \frac{\mathbf{O}_c\mathbf{P} \cdot \mathbf{T}_i\mathbf{O}_c + \|\mathbf{PW}\|\left(1 - \|\mathbf{T}_i\mathbf{O}_c\|^2\right)}{\|\mathbf{T}_i\mathbf{O}_c\|^2} \end{aligned} \tag{5-35}$$

Where $\|\mathbf{PW}\|$ is defined in Equation 5-33.

To summarise, the occlusion penalty is the multiplication of two sigmoid functions, one for the horizon plane and one for the occlusion cone:

$$s_{occl}(\mathbf{P}, \mathbf{n}_{\text{horizon}}, \mathbf{h}_{\text{horizon}}, k_{cone}) = s(\mathbf{P}, \mathbf{n}_{\text{horizon}}, \mathbf{h}_{\text{horizon}}) s(k_{cone}) \tag{5-36}$$

The equations from this chapter map a position in such a way that the quantification of information does not depend on binary constraints and measures occlusion of an obstacle. Applying a score for each triangle, as defined in Equation 4-20 and transforming using Section 5-1, now becomes a continuous function:

$$s_i(\mathbf{P}) = A(T_i) \left(\sum_{j=1}^{8} s_{constr}(\mathbf{P}, \mathbf{h}_j, \mathbf{n}_j) + \sum_{q=1}^{n_O} s_{occl}(\mathbf{P}, \mathbf{n}_{horizon,q}, \mathbf{h}_{horizon,q}, k_{cone,q}) \right) \tag{5-37}$$

Where the first term in the sum penalises being positioned away from the optimal location, the second sum penalises occlusion of obstacles and n_O is the amount of obstacles. Now, summing over each triangle results in an information 'penalty' and can be applied as a cost function for the MPC.

Chapter 6

Experimental setup

In order to determine the performance and limitations of the system, several simulations are performed in different environments, both free of obstacles and cluttered. Two objects are inspected, a simple geometry and a more complex real-life object, the tail of an aircraft.

First, the hardware, software, libraries and parameters which are used to come to the final implementation of the method are discussed. This is followed by the experiments, showing the robot behaviour during inspections and allow for determining if the proposed method is indeed robust.

6-1 Implementation

All simulations are performed on Ubuntu 18.04 with a QuadCore Intel i7 CPU @ maximum clock speed of $4.6GHz$. All programming is done using Robotic Operating System (ROS) (Quigley *et al.* 2009) in C++. This allows for easy data generation by means of bagfiles, fast execution of efficient coding and is the main developing environment of Mainblades Inspections. Visualisation is performed by Gazebo GUI and graphs are generated using the plotting library Matplotlib for Python.

For implementing the controller, the open source library control toolbox (Giftthaler *et al.* 2018), or "optcon" is used to build the MPC. As a general rule of thumb, updating input for an MAV should be performed every $15 - 20Hz$. Therefore, the time step is set to $h = 0.05s$. For solving each optimisation problem, the High-performance interior-point-method (HPIPM) library (*High-performance interior-point-method*) is used and the yaw angle ψ is kept constant to speed up the optimisation. This solver is the only interface allowing constrained control. Each term in J, as per Equation 4-29, is differentiated by MatLab to obtain the Jacobian- and Hessian matrices for fast optimisation.

Ceres-solver, developed by Google (*Ceres Solver*), is implemented for off-line optimisation of the information gain (mentioned in Section 6-2). This solver calculates the optimal location with by formulating a non-linear least squares problem and using gradients to find a solution.

The maximum iterations allowed is set to max_iter = 5000 to optimise $I(\mathbf{P})$ for Algorithm 2. Ceres uses non-linear least squares combined with automatic differentiation.

The MAV is modelled as a DJI M100 with drone radius $r_{\text{drone}} = 0.375m$, mass $m = 2.50kg$. The minimal distance before collision penalties come into play is $a = 1.00m$. Regarding the sigmoid functions, the optimal distance is set to $d_{opt} = 0.020$, the minimum- and maximum distances are $d_{min} = 1.00m$, $d_{max} = 6.00m$. The weights are set as:

$$\begin{aligned} w_{height} &= 5.0 \\ w_{incidence} &= 1.0 \end{aligned} \tag{6-1}$$

The weights for the cost function are:

$$\begin{aligned} w_I &= 1.0 \\ w_O &= 5.0 \\ w_M &= 1.0 \end{aligned} \tag{6-2}$$

Weights for the mesh are set to 1.0, because the robot is generally close to multiple triangles at once, effectively increasing the penalty without having a high weight.

Lastly, a check to determine if the MAV is hovering stationary is implemented as follows. The difference of the current state and one second ago is calculated and must be smaller than some ϵ. This is a simple way of checking the settled state of the system and is mathematically written as follows:

$$\|\mathbf{x}_k - \mathbf{x}_{k-d}\| > \epsilon \tag{6-3}$$

Where d is the first integer such that time $k - d$ is 1 second before k. Choosing $\epsilon = 0.1$ will make sure the MAV practically hovered around the same location for one second.

6-2 Baselines

Line 8 of Algorithm 1 (moving to the next viewpoint) is a function that can be solved in different ways. To determine if the use of an MPC outperforms the current state-of-the-art, three methods will be considered. Differences in methods will be minimised by using exactly the same MPC implementation (see Section 6-1) and only altering the types of optimisation. Furthermore, a penalty on the input is added to the cost function for smoother movement. It is defined in the form of a quadratic cost function:

$$J_u(\mathbf{u}_k) = \mathbf{u}_k^T R \mathbf{u}_k \tag{6-4}$$

With R being a diagonal weighting matrix, used with the following values:

$$R = \text{diag}(\begin{bmatrix} 0.3 & 0.3 & 0.01 \end{bmatrix}) \tag{6-5}$$

6-2-1 View global planner

The baseline method will simply execute the path obtained by the view global planner from Section 2-1. Each viewpoint ν is directly passed to a cost function J_{p2p}, used for Point-to-Point (P2P) control, which is defined as:

$$J_{p2p}(\mathbf{x}_k) = (\mathbf{x}_k - \nu)^T Q(\mathbf{x}_k - \nu) \tag{6-6}$$

Where ν is the location of the next viewpoint. This cost function will drive the states to ν, keeping in mind that $\nu \in \mathbb{R}^6$ with $\dot{x}, \dot{y}, \dot{z}$ set to zero. Q is a weighting matrix with the following values:

$$Q = \text{diag}(\begin{bmatrix} 10.0 & 10.0 & 10.0 & 10.0 & 10.0 & 10.0 \end{bmatrix}) \tag{6-7}$$

Algorithm 2, without execution of Line 2, is used to perform an inspection.

6-2-2 Off-line optimisation

The second method, as shown in Algorithm 2, solves the cost function in an off-line manner and then use a quadratic cost function for point-to-point control. This approach is closely related to the state-of-the-art mentioned in Chapter 3, e.g. for $3D$ exploration by Bircher *et al.* 2016a and for surface inspection by Bircher *et al.* 2016b and Lin *et al.* 2017.

The cost function contains lots of local minima. Consequently, Ceres often finds a non-optimal solution. In practice, adding a bit of noise to the obtained location allowed Ceres to iteratively obtain a better solution. Convergence is determined by checking whether the solution fails to improve ten consecutive times in the direction of each axis.

6-2-3 Joint optimisation

The third method utilises an MPC to both optimise the cost function and move at the same time. The main difference with the off-line method is that the MPC will use the cost function defined by Equation 4-30 to optimise while moving. This method incorporates Algorithm 3 to perform an inspection.

Algorithm 2 Off-line movement

1: **procedure** FindNextViewpoint
2: $\nu \leftarrow$ Optimise(J)
3: MPC $\leftarrow$ AddQuadraticCost($J_{p2p}(\nu)$)
4: **while** $\|\mathbf{x}_k - \mathbf{x}_{k-d}\| > \epsilon$ **do**
5: $[\varphi, \theta, \psi, F_{thrust}] \leftarrow$ SolveMPC()
6: ApplyAttitude($\varphi, \theta, \psi, F_{thrust}$)
7: $\mathbf{x}_k \leftarrow$ MeasureState()

Algorithm 3 On-line movement

1: **procedure** FindNextViewpoint
2: MPC $\leftarrow$ AddCost(J)
3: **while** $\|(\mathbf{x}_k - \mathbf{x}_{k-d}\| > \epsilon$ **do**
4: $[\varphi, \theta, \psi, F_{thrust}] \leftarrow$ SolveMPC()
5: ApplyAttitude($\varphi, \theta, \psi, F_{thrust}$)
6: $\mathbf{x}_k \leftarrow$ MeasureState()

6-3 Experiments

In order to verify the desired thesis objective, a couple of experiments have been performed. These will indicate whether the algorithm guarantees full coverage after the inspection while avoiding collision.

Two different worlds are considered. First, a simple cube will be inspected, followed by a more complicated mesh, the tail of an aircraft. These are depicted in Figures 6-1 and 6-2, respectively. Every object its location is known, as well as its size. Each object will also be inspected without obstacles, to demonstrate a baseline for complete inspections.

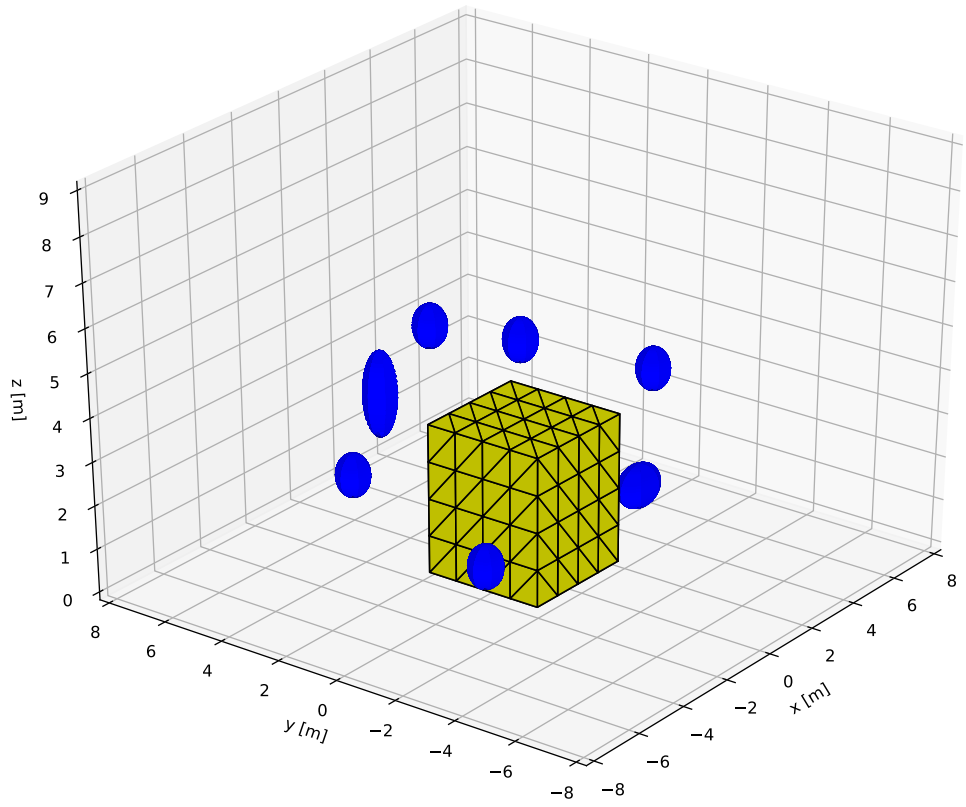


Figure 6-1: The world exhibiting a cube for inspections, cluttered with obstacles.

The complete inspection is shown (in Chapter 7) as percentage of A_{gp} covered over time, for each of the methods. Besides a complete inspection, four other experiments are performed to determine the performance of the proposed inspection method.

First of all, the derived function mapping position to information gain is checked for consistency. Before proceeding, it is very important to verify this function, which can be done by checking the resulting scores.

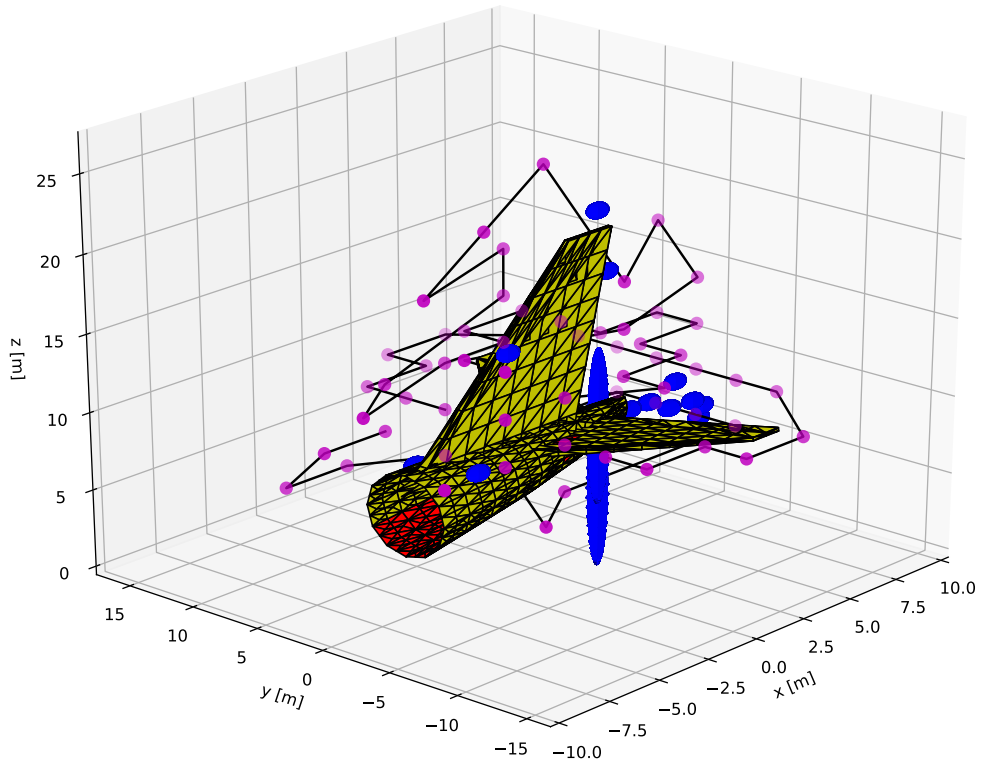


Figure 6-2: The world exhibiting a cube for inspections, cluttered with obstacles.

Second, the cost function becomes increasingly complex for each added triangle to the inspection problem. Therefore, the calculation times for each MPC iteration are checked to determine the scalability of the method. A boxplot is obtained by 25 consecutive calculations of the MPC for 5 different runs, resulting in 125 data points.

Third, settling behaviour indicates whether the controller is stable and tuned well. By running the simulation 5 times and letting the MAV continue for $t = 21s$, data is obtained by taking the Root mean squared error (RMSE) over time.

Fourth, the collision avoidance needs to be determined. For a single complete inspection, the minimal distance from the centre of the MAV to the obstacles and mesh are plotted over time. When this distance falls below the radius of the MAV, collision occurs.

The experiments are summarised in Table 6-1 along with the reason why each experiment is performed.

Table 6-1: The experiments and why they are performed.

Experiment	Reasoning
Information gain function verification	The information gain must show the expected cost for the algorithm to work.
Calculation times comparison	This describes the feasibility and scalability of using MPC for inspection problems with increasing size.
Settling times comparison	Settling times show limitations on how the robot behaves around optimal locations.
Distance to obstacles	Collision avoidance needs to be determined.
Performing full inspections	Completing inspections with- and without obstacles indicate whether the proposed method is indeed more robust.

Chapter 7

Results

This chapter will show and discuss the most important graphs supporting the main objective of the thesis and lay the foundation for the concluding chapter. Additional graphs are shown in Appendix C and referred to with their respective subject.

First, the information gain function is verified. The true optimal inspection location must be close to the evaluations of the information gain. By checking several scenarios, it can be determined if the information gain resembles inspection. Second, the solving times for each MPC iteration is looked at to determine scalability. Then, the settling times and distances to obstacles are considered to determine the stability and robustness. Lastly, complete inspections for two worlds, with- and without obstacles are performed to check the quality of the method.

7-1 Information gain function evaluation

Visualising the information cost is covered in this section. Each image, unless otherwise stated, consists of four graphs. The top left function shows the environment, where green triangles are observed and red are not. The red dot indicates the position for which the actual scores are represented in the other three graphs. Each of these three shows a different two-dimensional plane and its respective score, effectively visualising the score associated to a three-dimensional space.

These images only contain the information function of the first inspection problem (corresponding to the top viewpoint of Figure 4-2). Moreover, the spacing $\Delta = 0.05m$ in every direction to obtain clear graphs of the overall function.

The function evaluation for a single triangle and without obstacles is illustrated in Figure 7-1, where the optimal position $\mathbf{P}_{opt}$ and centre of mass of the triangle $\mathbf{T}_{c,m}$ are:

$$\mathbf{P}_{opt} = \begin{bmatrix} 0.6 \\ 0.6 \\ 4.6 \end{bmatrix} \quad \mathbf{T}_{c,m} = \begin{bmatrix} 0.62 \\ -0.62 \\ 3.43 \end{bmatrix} \tag{7-1}$$

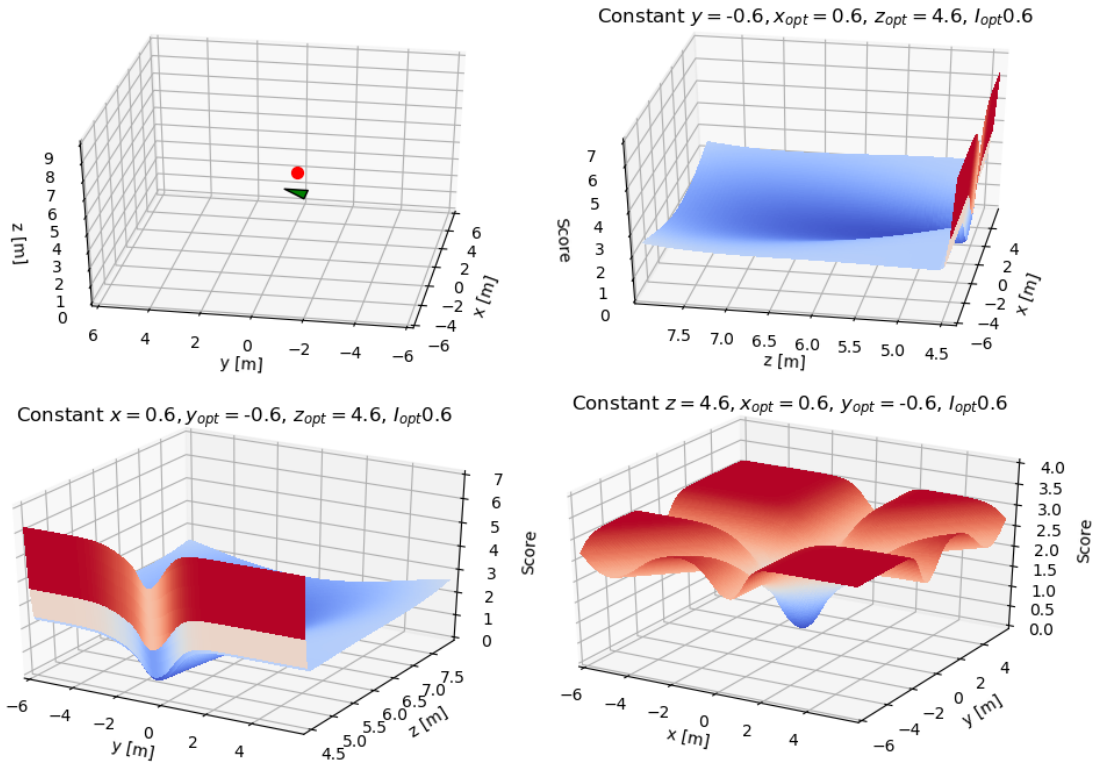


Figure 7-1: The score $I(\mathbf{P})$ for a single triangle on the optimal position.

Following the example given in the first paragraph of 5-3-2, the optimal height is found to be at $1.1m$ above the triangle. This leads to a distance off the 'true' optimal location of about $0.066m$.

The lower right graph indicates three valleys, each corresponding to the optimal lines derived in Section 5-3. The top right and lower left graphs are roughly the same. Each indicates the cost becomes high when $z < 4.5$, where the minimal distance height constraint becomes active. The influence of the maximum height constraint is barely visible. This is due to the slope being very small, because the sigmoid function stretches over several metres, instead of centimetres. The round-off is most likely the reason for the larger deviation in height for $\mathbf{P}_{opt}$.

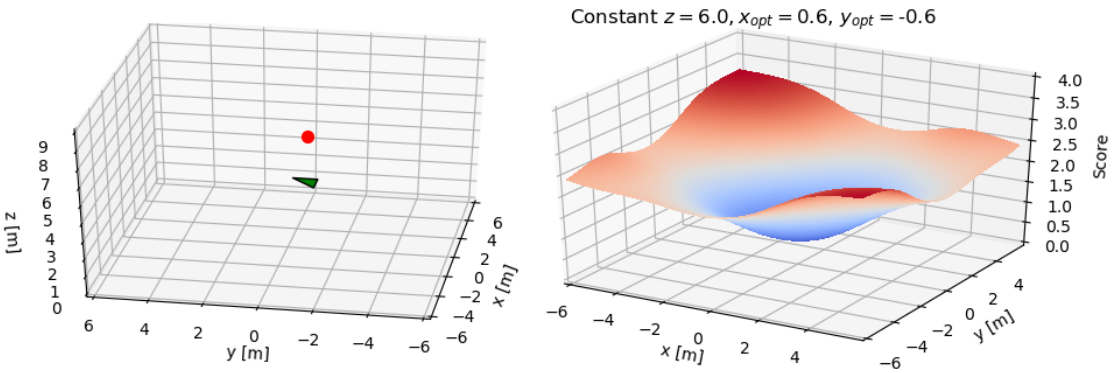


Figure 7-2: The score $I(\mathbf{P})$ for a single triangle on a higher position.

A red dot located straight above the one of Figure 7-1 is shown in Figure 7-2. The graphs where the x- and y-axis were constant are exactly the same, therefore omitted from this Figure. As expected, the steep valleys have flattened due to increased altitude, but the optimal location stays directly above the centre of mass of the triangle.

The case when an obstacle becomes present is shown in Figure 7-3. To illustrate the evaluation, the desired location is at:

$$\mathbf{P} = \begin{bmatrix} 0.6 \\ 0.6 \\ 5.0 \end{bmatrix} \tag{7-2}$$

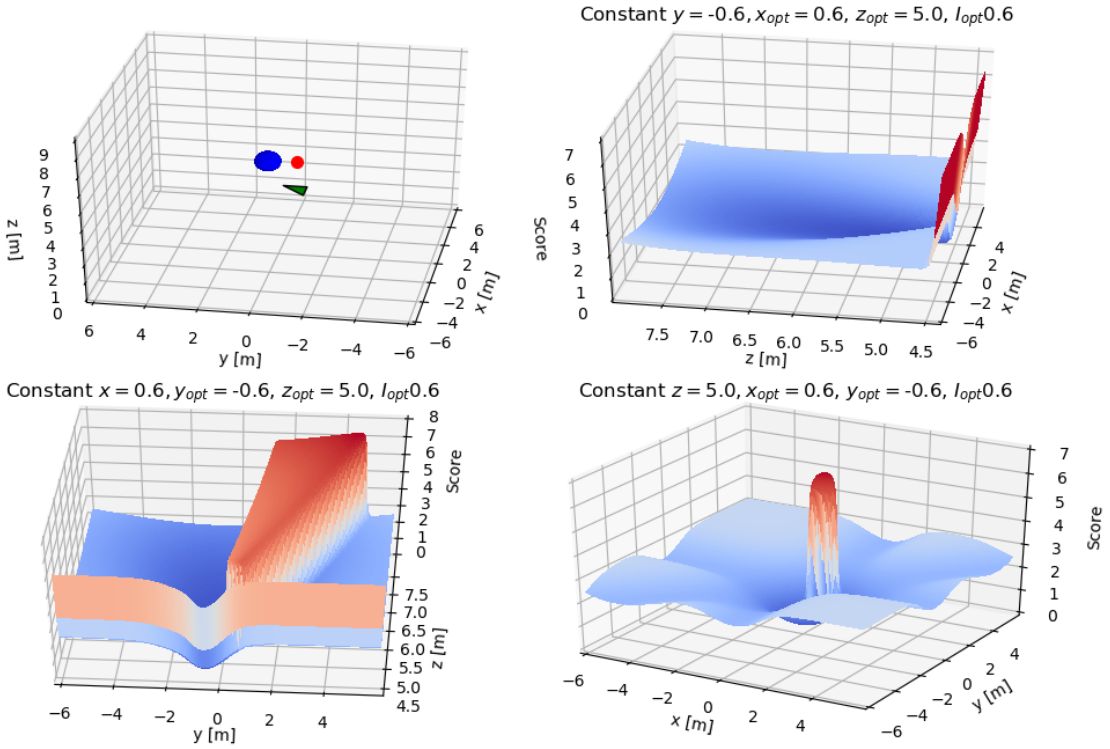


Figure 7-3: The score $I(\mathbf{P})$ for a single triangles when an obstacle is present.

The hill in the lower left graph indicates occlusion by the obstacle. When passing beyond the $y = 0.5m$ mark, the obstacle comes into play, increasing the penalty given for that location. The line of sight is also verified as the diagonal part of the same hill. When y is held constant, no obstacle is present in that direction. Hence, the graph is exactly the same as the top right graph from Figure 7-1. Lastly, when looking closely at the bottom right graph, a hill with small diameter can be seen. This corresponds to the location of the obstacle causing occlusion.

More figures can be found in Appendix C, where more triangles have been added to the information gain function.

7-2 MPC solving times

The calculation times for each MPC iteration, plotted against an amount of triangles per problem using boxplots, can be found in Figures 7-4(a) and 7-4(b), without and with obstacles, respectively. A linear pattern for increasing solving time per calculation is observed. The time it takes Ceres to solve problems is shown in Appendix C.

Figure 7-4(a) shows that around 20 triangles, the calculation time consistently exceeds the timestep of $0.05s$, effectively prolonging the execution of calculated inputs. In turn, this decreases overall stability of the flight. At 32 triangles, the mean calculation time becomes $0.125s$.

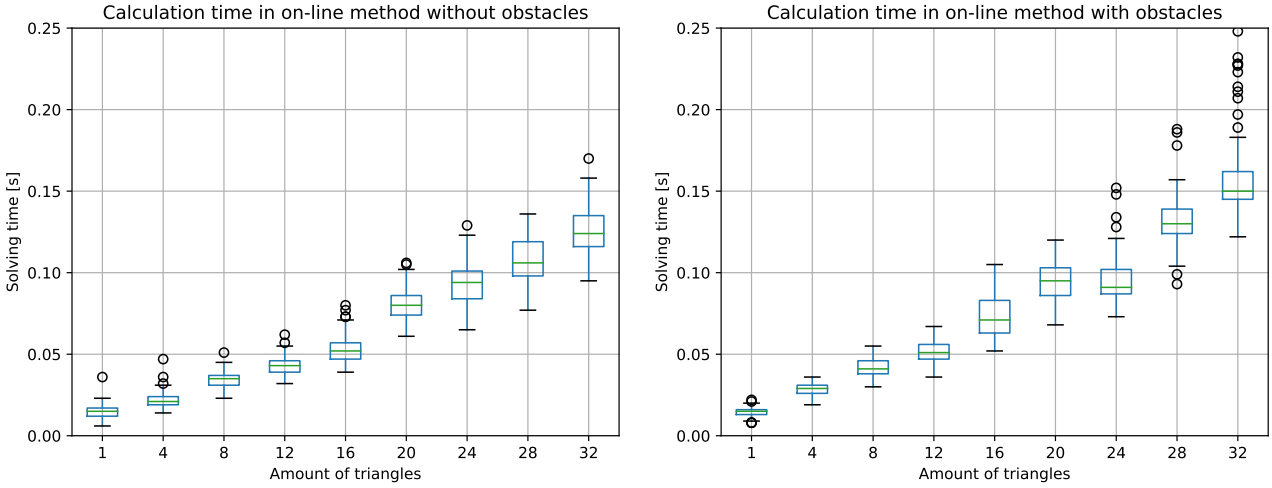


Figure 7-4: The calculation times for each iteration of the MPC per size of the problem, expressed in triangles, without obstacles on the left and with obstacles on the right.

When obstacles come into play, the cost function becomes more complex, leading to longer calculation times. Complexity is measured in amount of obstacles, unaffected by size, the same as triangles. Figure 7-4(b) illustrates that now, in the presence of a single obstacle, only 16 triangles are needed to consistently bring the calculation time above $0.05s$. For 28+ triangles, the system is unable to converge to an optimal position, even when applying large weights to velocities such that the MAV moves very slow to overcome the input delays.

7-3 Settling times

Settling times show the MAV move to the next viewpoint and trying to maintain position. The target used for the graphs in this section are the locations at which images were taken by the algorithm. By taking the RMSE of 5 trajectories, the resulting graphs are obtained. The settling time for the P2P method is illustrated in Appendix C.

Figure 7-5 shows the settling time for different amounts of triangles. As expected, when the inspection problem consists of 16 triangles, the settling time is smooth with little to no error.

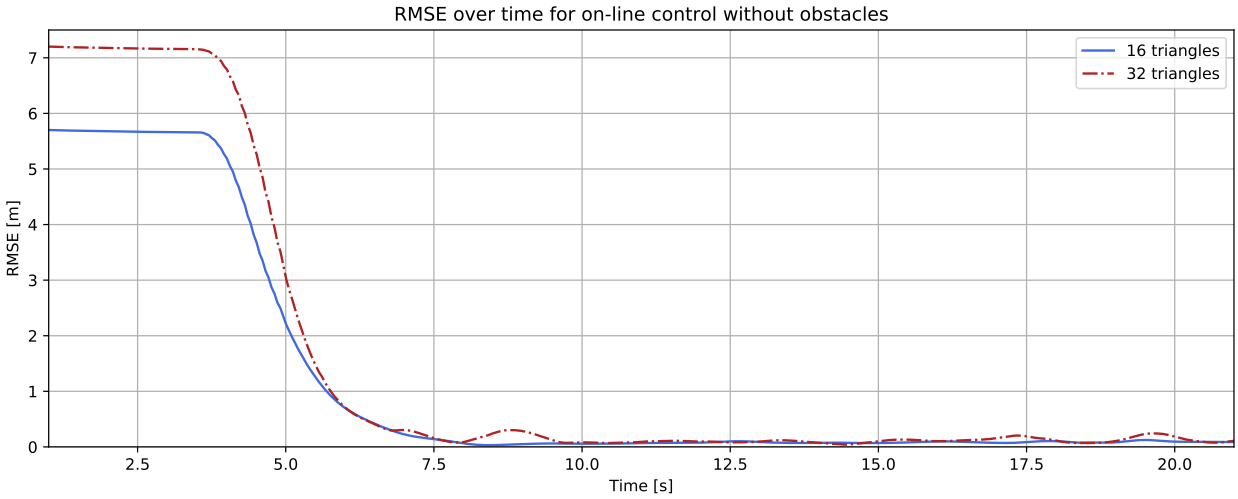


Figure 7-5: Settling times for the on-line method without obstacles.

The average picture time is at $8.6s$, but the MAV is still settling a bit more. The small offset can be explained by the way the system checks when the robot is in a settled state (as per Equation 6-3). Increasing the problem to 32 triangles shows slightly more ringing. This indicates that the robot stays within a small region around the target, but does not truly converge. Moreover, the condition for settling was not met for all datasets and the target chosen is the settling point for when no intermediate cost is applied in the cost function. This is discussed in the next paragraph.

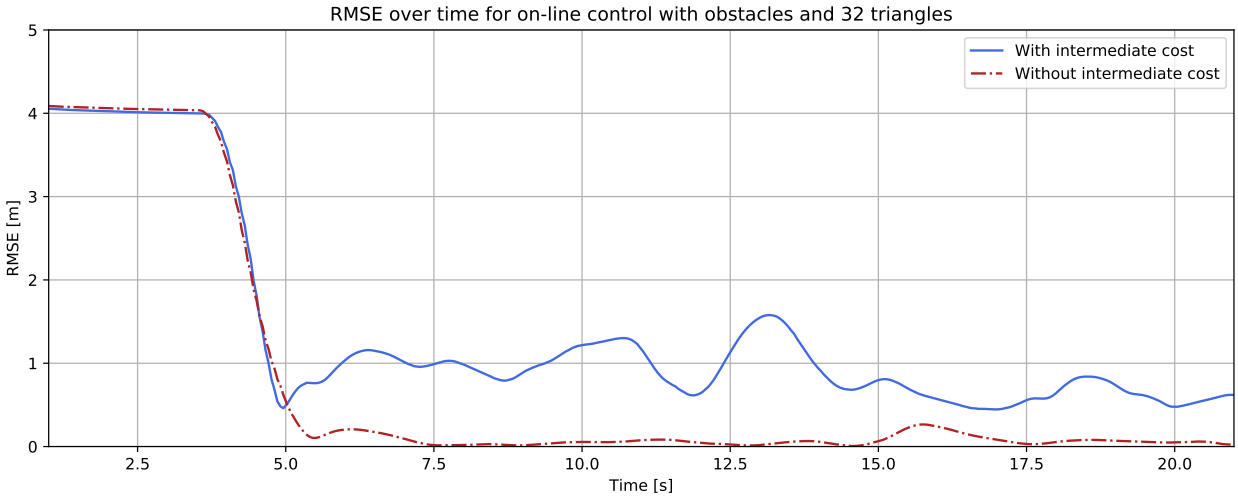


Figure 7-6: Settling times for the on-line method with an obstacle present.

Figure 7-6 shows the settling time for 32 triangles when an obstacle comes into play. Without obstacles, the MAV showed reasonable stability. However, this is not the case anymore and the MAV is having a real hard time even staying within $1m$ of the target, as shown by the blue line. Dismissing the intermediate cost of the information cost for the MPC reduces the

computation times significantly enough to produce a fast and fairly consistent settling time. By doing this, the resulting settling time is the one shown in red. This clearly improves the stability of the MAV, such that inspection problems with obstacles become feasible for larger amounts of triangles. The targets acquired for the cost functions without intermediate costs are used for this figure and the red line of Figure 7-5.

7-4 Complete inspections

For complete inspections, the area covered is expressed as a percentage over time. The setting in which the inspection takes place is the environment of Figure 6-1 combined with the path shown in Figure 2-3. First the quality will be discussed, followed by the distances to obstacles. The simulation of the tail of an aircraft is discussed in Section 7-4-3 and the distances to obstacles for the global planner method is discussed in Appendix C.

7-4-1 Quality of the inspection

Figure 7-7 shows inspections of the cube and without obstacles and Figure 7-8 shows inspection with obstacles. The on-line method is performed without intermediate costs. In other words, the information cost is only applied to the final state over the prediction horizon of the MPC.

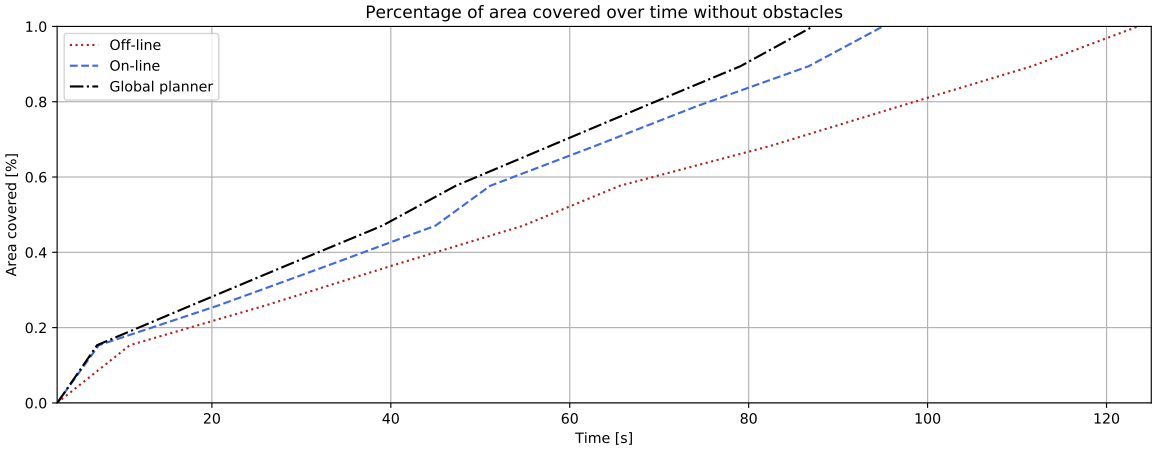


Figure 7-7: The accumulated area percentage of the complete inspection of a cube over time, without obstacles.

Without obstacles, the fastest way to perform inspections is to directly pass the viewpoints from the global planner to the MPC, resulting in a fully covered cube in 87.2s. The on-line method seems to be slightly slower, with 95.0s, but much faster than the off-line method, completing the inspection in 123.5s.

Obstacles, however, interfere with the original route, as can be seen in Figure 7-8. The simulation performed with the global planner is partly occluded by obstacles and therefore, does not complete the inspection. Both on-line and off-line methods handle the obstacles and

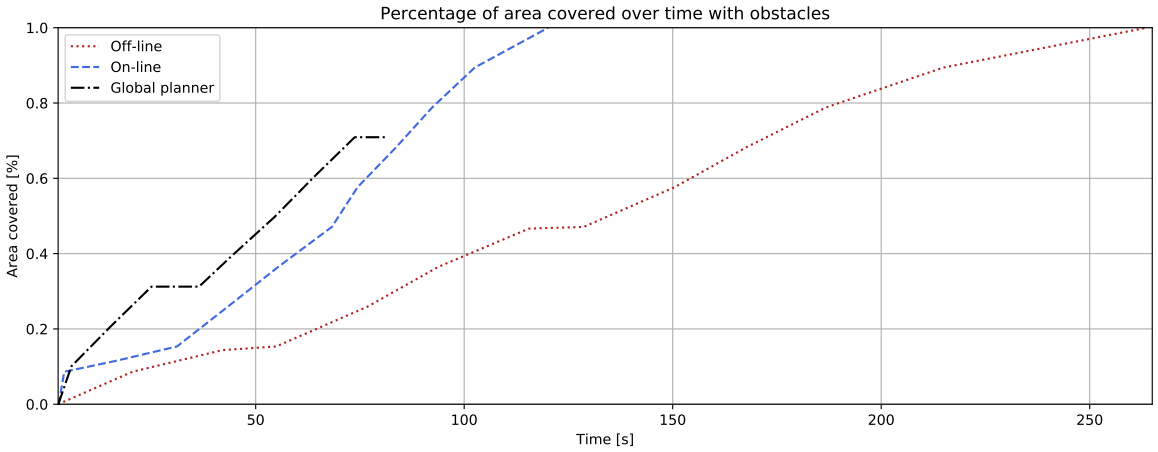


Figure 7-8: The accumulated area percentage of the complete inspection of a cube over time, with obstacles.

reach 100% area covered of M_{gp}, determined by the global plan. It is worth noting that the off-line method takes more than twice as long, compared to the on-line method, completing the inspection in $263.7s$ and $120.14s$, respectively.

While these simulations complete the inspection, it is not a guarantee, as the algorithm sometimes misses a triangle for the same inspection problem. This is most likely due to the initial solution used for the methods ending up in local minima.

The quality of the solution can be discussed by checking the score for each triangle over the complete inspection. Using Equation 4-20, describing the score of a triangle as the distance times incidence angle, the scores over time for an inspection can be plotted in a single graph, where each viewpoint is appointed a score and the accumulation is interpolated linearly.

Each line in Figure 7-9 represents the score accumulated over time without obstacles. The differences in score between the off-line and on-line method can be explained by the difference in cost functions. For the on-line method, penalties for input and velocities are considered, but not for the off-line method. This causes a displacement in the optimal location.

The global planner performs the worst of the three methods. This can be explained by the discretisation of the global planner algorithm. The path consists of less optimal viewpoint locations, which are mitigated by the off-line and on-line methods, causing lower scores for the global planner method. The final scores for the methods are given in Table 7-1.

Graphs including obstacles in the scores are shown in Figure 7-9(b). Again, the global planner performs the worst. What is interesting to note is that both the off-line and on-line methods score almost equally to their non obstacle-based counterparts from Figure 7-9(a), while the global planner its score is reduced. This implies the quality of the solution is preserved, but the inspection duration increases.

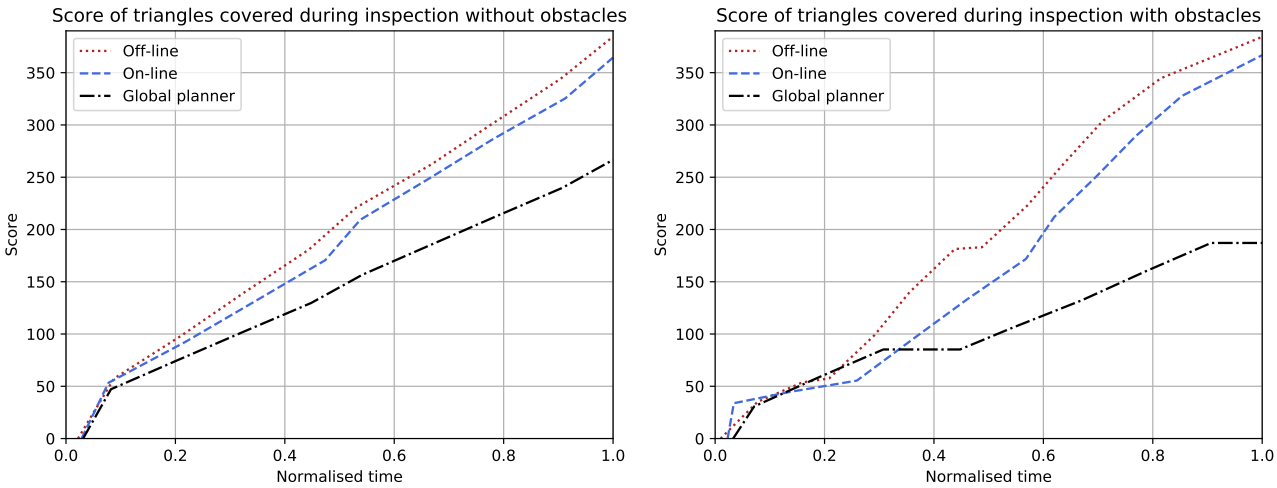


Figure 7-9: The accumulated score (defined by Equation 4-20) over a complete inspection of a cube, without obstacles on the left and with obstacles on the right.

Table 7-1: The total accumulated scores over time for an inspection.

	Method	Score	Time taken
No obstacles	Global planner	266.64	87.19s
	Off-line	384.52	123.52s
	On-line	364.48	95.00s
Obstacles	Global planner	187.11	87.19s
	Off-line	384.23	263.68s
	On-line	366.78	120.14s

7-4-2 Obstacle distances

The next figures show the distance to the obstacles in the environment. Each graph shows three lines, corresponding to the minimum distance of all obstacles, minimum distance to the triangles of object to inspect B and the radius of the bounding box for the MAV.

Figure 7-10 shows the distances for the off-line method. During a complete inspection, the centre of the MAV comes within $0.5m$ three times. These are at roughly $112s$, $148s$ and $163s$, with minimal distances of $0.444m$, $0.448m$ and $0.353m$, respectively. Recalling the drone radius was set at $0.375m$, the MAV very slightly collides with an obstacle. Even though the algorithm discourages colliding, the proper way to avoid collision would be to add hard constraints or increasing the activation distance and weight for the distance penalties.

The distances to obstacles for the on-line method is shown in Figure 7-11. Here, the robot comes within $0.5m$ of an obstacle twice at $81s$ and from $121s$ onward at minimum distances of $0.272m$ and $0.487m$, respectively. Again, the robot collides with an obstacle during inspection.

In both cases, however, the robot always operates from a safe distance from the mesh. This is also expected, because the path computed by the global planner is clear of obstacles. Therefore, obstacles within the environment are more important to consider.

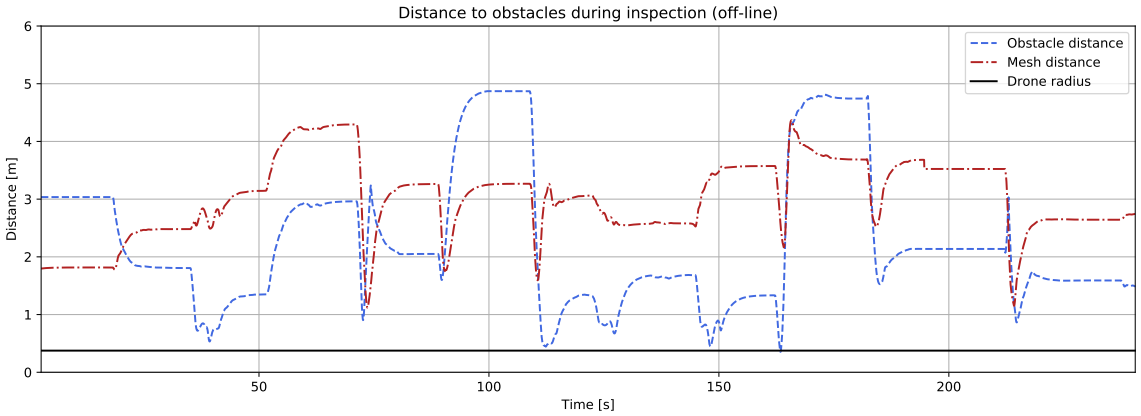


Figure 7-10: Distances to obstacles for off-line method. The distance to an obstacle is defined as the centre of the MAV to the edge of the obstacle.

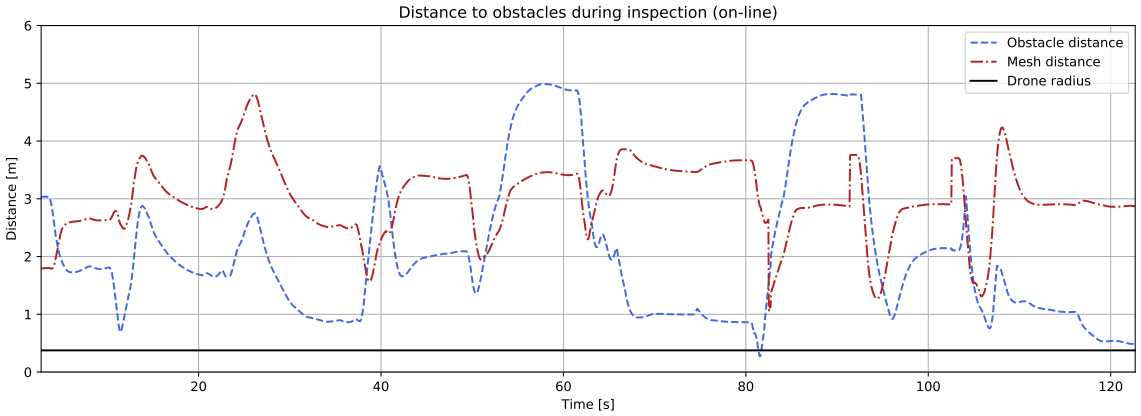


Figure 7-11: Distances to obstacles for on-line method. The distance to an obstacle is defined as the centre of the MAV to the centre of the obstacle.

7-4-3 Aircraft tail

The simulations with an aircraft tail did not result in any useful data. Due to the first viewpoint covering 186 triangles, even the point-to-point control method, without obstacles, took more than $1.2s$ per iteration to find a solution. Each of these triangles have their own distance cost, resulting in over 3600 terms, leading to long calculation times. For this reason, further simulations are dropped as each method increases the mathematical complexity even more.

Chapter 8

Conclusion & recommendations

This final chapter concludes the thesis and reflects on what goals are, or are not, achieved. It reflects on the performed work and provides recommendations for future directions.

8-1 Conclusion

This research aimed to develop an on-line algorithm capable of autonomous inspection, while rejecting obstacles that are encountered on-line. Based on simulations in both cluttered and free environments, it can be concluded that the proposed method improves the solution of the global planner, at the cost of being slower. However, the current implementation does not guarantee complete coverage, as well as collision free inspection. The results indicate a computationally heavy algorithm, incapable of stable flight without making a concession in the structure of the cost function for robust inspection.

The main objective stated in Chapter 1 is:

> *"Design an on-line algorithm that is able to perform visual inspection of a 3D mesh object, collision-free in an occluded environment, implementing a Model Predictive Controller and guaranteeing full coverage."*

In order to try and solve this problem, different steps were taken as an approach. Current state-of-the-art often use precomputed paths to traverse, potentially encountering obstacles that are unaccounted for. When these methods are not used, exploration-based approaches, combined with off-line methods such as TSP solving, result in a lot of standstill time for the MAV without any form of coverage guarantee. However, one thing they all have in common is a means to quantify information. By combining both approaches and coming up with a way to drive the robot to the most informative location using an MPC, an on-line algorithm is developed that fulfils the main objective.

Submodular set functions describe problems with diminishing returns. A very useful property is the fact that greedy optimisation can be used to obtain performance guarantees. With the

analogy of inspection problems to submodular set functions, the solution to a cost function for the MPC acts as a node in the greedy solution of the inspection, gaining guarantees.

By defining a space for each triangle on the mesh for which it is observable, the amount of information can be translated from position to a number. This space consists of eight linear constraints per triangle, approximated by sigmoid functions. Applying two more constraints, when obstacles are present, steers the robot towards non-occluded areas. What follows is a cost function, built from the part of the mesh a single viewpoint of the global path is able to observe. Going through each viewpoint will result in an iterative procedure, contributing to the complete inspection.

While the constructed custom MPC works for smaller inspection problems, adding triangles or obstacles will result in unstable behaviour. In order to resolve this issue, the intermediate cost associated with the information gain needs to be dismissed. This reduces the computational load to acceptable levels and avoids delayed inputs. The resulting method is less elegant, but works during simulations of the cube. Simulations with the tail of an aircraft resulted deemed not viable. Due to many triangles per viewpoint, obstacle avoidance for triangles caused too long calculations times, even for the point-to-point control method.

An assumption was made stating when sufficiently penalising obstacle distance, no collision occurs. During simulations, it was found the current implementation does not adhere to this. The collisions measured were barely over the limit, encouraging more penalties or, when the assumption is false, adding hard constraints to the MPC in order to avoid this.

The proposed method does indeed perform better than simply executing the path obtained by the view global planner. Being slower is the price to pay for avoiding occlusion of obstacles, obtaining more coverage of the object to inspect and preserving the quality of the solution. However, complete inspections are not guaranteed, because in practice the robots sometimes gets stuck in local minima, skipping one or several triangles.

8-2 Recommendations

Based on this work, the primary limiting factor is the calculation time for each iteration. Future work could look into other methods in order to reduce computation times. Two distinct problems can be looked at.

First, the obstacle avoidance. Due to penalising closeness to triangles, the MPC for inspection problems with many triangles ($\mathcal{O}(10^2)$) becomes unstable. Assuming research has come up with different solutions to incorporate mesh distances into an MPC, this would enable larger scaled inspection to be solved.

Second, information gain. Obstacle occlusion hits the performance hard. While the visibility of triangles is relatively complex, constructed by eight sigmoid functions, adding a single obstacle potentially cuts the size of the inspection problem in half. Replacing this term with a new, less complex, solution, performance might be improved a lot. A possible direction for this specific issue could be to incorporate a LIDAR sensor, where the generated point cloud is fitted on the mesh.

In a more general sense, future research can be performed with more complicated structures. Instead of using static objects, considering non-rigid obstacles, including self-occlusion, will

be very interesting. Finding a solution will provide a better understanding and execution of inspections. Furthermore, stepping away from deterministic approaches, such as this work, paves the way for a confidence of inspected area and a continuous stream of images by utilising stochastic approaches. Lastly, learning-based methods have the potential to reduce the on-line calculation times significantly, increasing the problem sizes.

Appendix A

Airviewtour: Viewpoint Selection and Tour Planning

Airviewtour: Viewpoint Selection and Tour Planning

Technical Report explaining the Coverage Path Planning algorithm for inspecting 3D structures

Nikhil Potdar
Cognitive Robotics, 3ME
Delft University of Technology
Delft, The Netherlands
N.D.Potdar-1@tudelft.nl

***Abstract*—This technical report describes the implementation of a Coverage Path Planning (CPP) algorithm for generating global viewpoint tours for obtaining aerial imagery for the inspection of a three-dimensional structure. The specific application vehicle case considered is for a Micro Aerial Vehicle (MAV) able to perform three-dimensional translation including yaw with an attached gimballed camera with yaw and pitch control. The viewpoint selection and tour planning is subject to photogrammetric and physical space constraints, and vehicle motion limitations.**

***Index Terms*—coverage, path, planning, viewpoint, selection**

I. Introduction

This is a technical overview of the *Airviewtour* algorithm for performing automated global path planning for the inspection of a three-dimensional orientable, closed-form structure using a mobile aerial platform.

II. Preliminaries

A. Triangular Mesh Representation of 3D Models

Viewpoint selection and planning is performed in a virtual 3D space where the object to be inspected is encoded as a 3D triangular mesh based model. Most common Computer Aided Design (CAD) models of 3D structures are publicly available in this format making this representation future-proof and applicable for wider use cases.

B. Viewing Frustum Camera Model

As commonly used in 3D computer graphics, a viewing frustum based camera model is necessary to compute the projection of 3D world objects onto 2D images. The model is used to filter the generated viewpoints based on whether certain photogrammetric (imaging) constraints are being satisfied based on the camera's specifications. All matrix arithmetic is performed in homogeneous coordinates simplifying the transformation operations.

Let A be the aspect ratio, c_n and c_f the near and far planes and α the vertical camera field of view in radians, then the camera projection matrix is given by,

$$P_c = \begin{bmatrix} p_{11} & 0 & 0 & 0 \\ 0 & p_{22} & 0 & 0 \\ 0 & 0 & p_{33} & p_{34} \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{1}$$

Delft University of Technology, Delft, The Netherlands, Mainblades Inspections, The Hague, The Netherlands

where $p_{11} = 1/\left(A\tan(\alpha/2)\right)$, $p_{22} = 1/\tan(\alpha/2)$, $p_{33} = (-c_n - c_f)/(c_n - c_f)$ and $p_{34} = 2c_f c_n/(c_n - c_f)$ [1]. These camera properties are schematised in Fig. 1.

The location and orientation of the camera in the world space is described by the camera transformation matrix, Fig. **??** shows the camera pose in the world frame.

Let $\boldsymbol{p_c} \in \mathbb{R}^3$ be the camera's global position and ϕ_c, θ_c and ψ_c be the roll, pitch and yaw rotations. The camera's orientation is given by the 3D rotation matrix comprising of starting axial rotations,

$$R = R_z(\psi_c)R_y(\theta_c)R_x(\phi_c) \ . \tag{2}$$

The camera transformation matrix is then defined by

$$T_c = \begin{bmatrix} R^{-1} & -\boldsymbol{p_c} \\ 0 & 1 \end{bmatrix} \ . \tag{3}$$

The transformation from the world to camera axis convention is given by,

$$T_{wc} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4}$$

The homogeneous transformation matrix from a world defined vector point to the image clip space is given by,

$$C_c = T_{wc}^{-1} \cdot P_c \cdot T_c \cdot T_{wc} \tag{5}$$

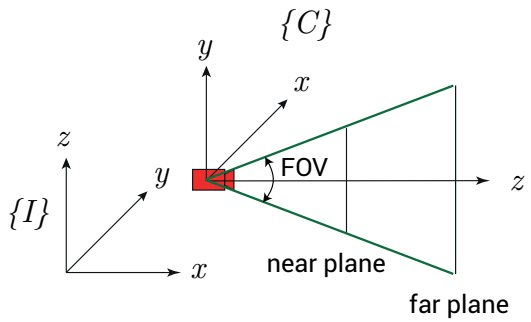


Fig. 1: Camera viewing frustum model with world $\{I\}$ and camera axis $\{C\}$ conventions showing the near, far plane and vertical Field of View (FOV). Camera viewing direction is aligned with the $\{C\}$ z axis.

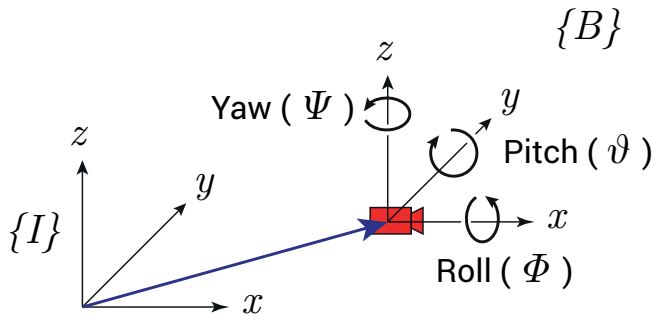


Fig. 2: World and camera body frames with camera pose shown.

where the homogeneous vector should be normalised. The image clip space, known as the Normalized Device Coordinates (NDC) space has components ranging from -1.0 to 1.0 and a world point is viewable if the transformation of the point by C_c results in a homogeneous vector with all components within this range. Transformation from clip space to the true screen space is performed by scaling the NDC components to the screen dimensions.

III. AIRVIEWTOUR ALGORITHM

A. Overview

The algorithm is intended for Coverage Path Planning (CPP) for a orientable, closed-form, 3D structure. The input to the algorithm comprise of;

- CAD 3D model of object to be inspected
- Workspace and vehicle motion constraints
- Desired photogrammetric requirements for images obtain from viewpoints
  - Full projection of triangle onto image plane according to camera model
  - Incidence angle of camera vector to mesh triangle normal vector
  - Minimum number of unique views observing a triangle fully

The algorithm generates a set Λ of ordered five degree of freedom, collision-free viewpoints with position $\boldsymbol{p}_c$ and camera gimbal yaw ψ_c, pitch ϕ_c such that

$$\lambda = [\boldsymbol{p}_c, \psi_c, \phi_c] \in \mathbb{R}^3 \times SE(2) \ . \tag{6}$$

The viewpoints are used to fully/partially observe the 3D model within the defined constraints and requirements. The pseudo-algorithm below gives an overview of the steps performed.

The algorithm is able to evaluate the coverage obtained and mark the surfaces that are covered and not covered subject to the requirements set out. The following sections explains the algorithm's steps in more detail.

Algorithm 1 Coverage Path Planning algorithm

1: **Data:** 3D Mesh
2: **Load** $\Lambda \leftarrow$ Generate regularly spaced grid with associated gimbal orientations within workspace giving n viewpoints denoted by λ which are ordered
3: $\Lambda \leftarrow$ Pre-filter Λ for each λ proximity to mesh
4: **for** Each $\lambda \in \Lambda$ **do**
5: $S(\lambda) \leftarrow$ Compute support of λ which is the number of full mesh triangles viewed with configuration λ, respecting the camera model and photogrammetric constraints
6: **end for**
7: $\Lambda \leftarrow$ Sorted list of all λ in descending $S(\lambda)$ order.
8: $\Lambda \leftarrow$ Greedily select viewpoints from Λ maximising the number of mesh triangles viewed and satisfying requirement for minimum number of unique views per triangle
9: $\Lambda \leftarrow$ Re-order Λ by solving a Travelling Salesman Problem (TSP) minimising the tour length, and changes in altitude between viewpoints while only considering collision-free edges

B. Mesh Definition

The mesh must be closed-form and orientable to allow the use of the developed algorithm. With orientable, it is required that each triangular face has an associated normal vector that faces away from the outer surface.

C. Viewpoint Generation

The workspace limits should be defined such that it encloses the bounding box of the mesh and represents a cuboid. A regular Cartesian grid with cubic cells is generated with a user specified edge dimension to create a cellular decomposition of the workspace.

The set Λ of n candidate viewpoints λ is defined at the locations of the grid vertices resulting in three Degrees of Freedom (DOF). Furthermore, a set of possible gimbal camera orientations are defined per viewpoint. The orientations are defined with all combinations of fixed increments in camera yaw and pitch, with associated maximum and minimum limits. This results in an additional two DOFs per viewpoint.

D. Pre-Filtering Viewpoints

The candidate viewpoints are pre-filtered according to several global criterion as described.

1) Mesh proximity: Let Ω represent the mesh object, and let $\min d_\Omega(\boldsymbol{p})$ be the minimum signed distance of any point $\boldsymbol{p}$ to any point on the mesh's surface. Let $d_{\min} \geq 0$ and $d_{\max}$ be the minimum and maximum distance a viewpoint may be from the mesh surface. Given a viewpoint $\boldsymbol{p}_{\boldsymbol{\lambda}}$ location, then looping through all viewpoints, it is removed unless the following condition is satisfied

$$d_{\min} \leq d_\Omega(\boldsymbol{p}_{\boldsymbol{\lambda}}) \leq d_{\max} \ . \tag{7}$$

With this condition all viewpoints contained inside the mesh are also removed. Following this filtering the n set of viewpoints λ is either

E. Filtering using Camera Model

The support of a viewpoint $S(\lambda)$ is the total number of unique triangles that are projected onto the viewable image space described by the camera model located and orientated according λ. The triangles viewed is identified using the process of raycasting whereby an array of regularly spaced infinite length rays is generated in the camera frustum originating at the camera origin. Collision of each ray with the mesh is checked to identify the triangles within the camera field of view. The rays are spaced such that at the rays intersection with the camera's far plane, the maximum separation between the points on that plane is a pre-defined value as shown in Fig. 3

Subsequent use of the camera model filters the viewed triangles based on camera properties. Expanding on Line 5 in Algorithm 1, the support is computed with Algorithm 2.

Figure 4 shows an example viewpoint with the associated viewed triangles.

F. Viewpoint Sorting and Selection

The viewpoints Λ are sorted in descending order of $S(\lambda)$, therefore, viewpoints with the highest support (most unique mesh triangles fully viewed) are first in the ordered set. Expanding on Line 8 in Algorithm 1, viewpoint selection is performed greedily on the ordered set Λ. Viewpoints that only contribute new information are selected by the Algorithm 3.

Following the selection, all viewpoints are parsed and the viewpoint set is the solution to the Coverage Sampling Problem (CSP) of the full Coverage Path Planning (CPP) problem.

G. Tour Planning

The second part of CPP is Multi-goal Path Planning (MPP); in the algorithm's implementation a Travelling Salesman Problem is solved based on the Lin-Kernighan Heuristic (LKH) [2] as implemented in the Google Optimization Toolbox [3]. An important assumption is that the straight line paths between viewpoints is only considered for defining the global path.

For the optimisation, the square cost matrix V containing the viewpoint-to-viewpoint pair costs v is generated. The cost terms are described in the subsequent sections.

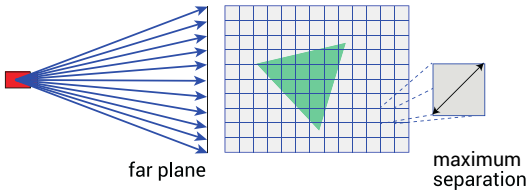


Fig. 3: Camera raycasting to check in discrete manner the mesh triangles observed with maximum separation between ray intersections with the far plane

1) Collision cost: For viewpoint λ_1 and λ_2, the straight edge joining them is checked for collisions with the mesh and

$$\begin{aligned} &\textbf{if} \quad \text{collision}, \quad && v_{\text{collision}} = w_{\text{collision}} \ , \\ &\textbf{else}, && v_{\text{collision}} = 0 \ , \end{aligned} \tag{8}$$

with $w_{\text{collision}} \gg 0$ such that if the edge is in collision, it will not be selected as a edge on the tour.

2) Distance cost: For viewpoint λ_1 and λ_2 with origins at $\boldsymbol{p}_1$ and $\boldsymbol{p}_2$ respectively, the weighted distance cost is defined by the Euclidean norm

$$v_{\text{distance}} = w_{\text{distance}} \|\boldsymbol{p}_1 - \boldsymbol{p}_2\| \ , \tag{9}$$

where w_{distance} is a positive weight.

3) Altitude change cost: For viewpoint λ_1 and λ_2 with z origin components z_1 and z_2 respectively, the weighted altitude change cost is defined by the Euclidean norm

$$v_{\Delta\text{altitude}} = w_{\Delta\text{altitude}} |z_1 - z_2| \ , \tag{10}$$

where $w_{\Delta\text{altitude}}$ is a positive weight.

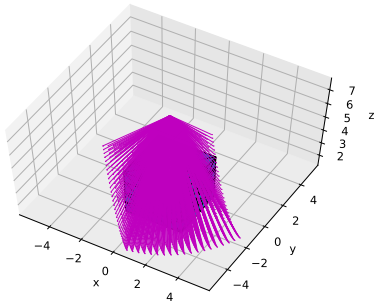
4) Total pair cost: The total viewpoint-to-viewpoint cost v is then

$$v = v_{\text{collision}} + v_{\text{distance}} + v_{\Delta\text{altitude}} \ . \tag{11}$$

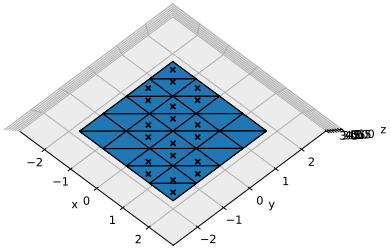
Solving the TSP with the cost matrix defined gives as output set of viewpoints re-ordered according to the sequence in which viewpoints should be executed to minimise the total tour cost. The final output is a global trajectory to be used for planning paths for a MAV type vehicle.

Algorithm 2 Camera Model based viewpoint filtering

1: **for** Each $\lambda \in \Lambda$ **do**
2: *Raycast* ← Generate a raycast array for the viewing frustum of the camera model
3: *Hit Triangles* ← Identify all mesh triangles hit by any ray of *Raycast*
4: *Hit Triangles* ← Filter *Hit Triangles* by requiring the angle between the camera direction vector and triangle normal vector to be less than the maximum incidence angle
5: *Viewed Triangles* ← Initialise empty set
6: **for** each *Triangle* ∈ *Hit Triangles* **do**
7: *Vertices* ← Project the *Triangle* vertices using the camera model
8: **if** all *Vertices* viewable **then**
9: *Viewed Triangles* ← *Viewed Triangles* + *Triangle*
10: **end if**
11: **end for**
12: $S(\lambda)$ ← Number of elements in *Viewed Triangles*
13: **end for**



Viewing frustum raycasting

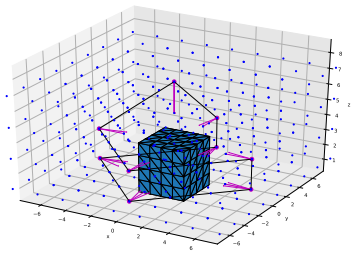


Top-down view showing viewed triangles marked with crosses

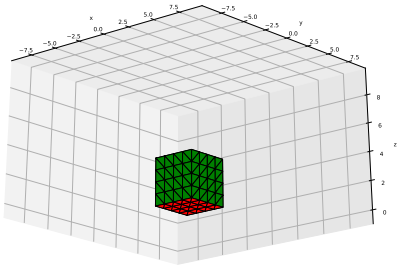
Fig. 4: Example viewpoint showing the triangles fully observed by the camera with respect to the photogrammetric requirements of the camera model.

Algorithm 3 Viewpoint greedy selection

$\bar{\Lambda} \leftarrow$ Empty set of viewpoints
Count $\leftarrow$ Initialise array with one zero entry per triangle of mesh to keep count of how many full triangle views from viewpoints are made
for each $\lambda \in \Lambda$ **do**
    for each *Triangle* $\in$ *Viewed Triangles* **do**
        if count of *Triangle* in *Count* $<$ Minimum full view of triangle required **then**
            Increment *Count* entries for all *Viewed Triangles*
            $\bar{\Lambda} \leftarrow \bar{\Lambda} + \lambda$
        end if
    end for
end for
$\Lambda = \bar{\Lambda}$



(a) Viewpoint grid with final generated tour



(b) Green and red triangles indicate viewed and unseen mesh surfaces respectively

Fig. 5: Tour of viewpoints generated to observe 5 out of 6 cube faces. Using 45° FOV, 2 m near plane and minimum distance, 5 m far plane, 6 m maximum distance, $16:9$ aspect ratio, 1 view per triangle.

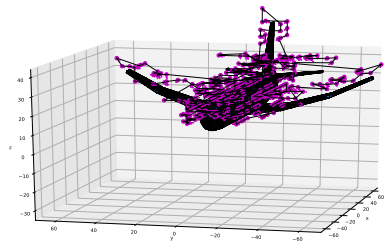
IV. Results

The generated tour to observe 5 out of 6 faces of a cube places on ground is shown in Fig. 5. Evidently, the bottom surface cannot be observed as the object is placed on the ground. A coverage of 82.4% of the cube's outer surface was achieved.

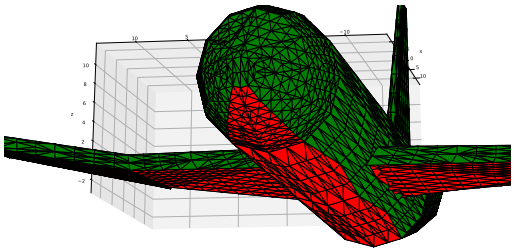
The generated tour that observes most of the aircraft's upper surface is shown in Fig. 6. A coverage of 67.6% of the full outer surface was achieved. Notice that the underside of the aircraft cannot be observed, this is in part to the limitations on the gimbal camera that is only able to observe at angles from the world horizontal down (limit pitch 0 to 90° and yaw 0 to 360°).

V. Future Improvements

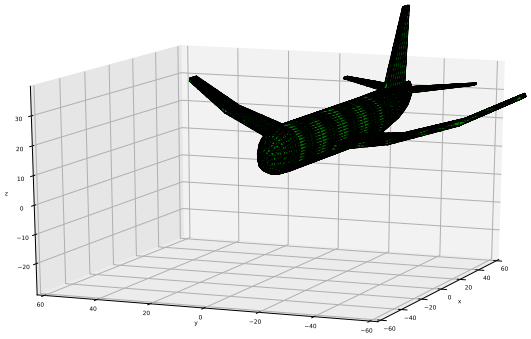
This is a working in progress technical document that is updated with changes and features to the algorithm. This section presents some possible future features for the algorithm that require possible research and engineering work. Specifically the features are geared towards the application of the algorithm for performing aircraft surface inspections.



(a) Final generated tour



(b) Green and red triangles indicate viewed and unseen mesh surfaces respectively



(c) Green and red triangles indicate viewed and unseen mesh surfaces respectively

Fig. 6: Tour of viewpoints generated to full aircraft where upper surface can be observed. Using $60°$ FOV, $1\ m$ near plane and minimum distance, $10\ m$ far plane and maximum distance, $16:9$ aspect ratio, 1 view per triangle.

A. Mesh Model Selection

The current code setup does not allow seamless selection of the mesh model to be inspected. A interface allowing this is useful for the end user as the algorithm generalises to any 3D structure, including different aircraft models, so plans can be generated for various aircraft.

B. Mesh Surface Selection

The current algorithm performs CPP on the entire mesh model. A procedure to select parts of the surface to be inspected is useful when only the fuselage or nose cone of an aircraft is to be inspected. This is useful when inspectors are aware of the high-risk aircraft surfaces when looking for certain damage.

C. Viewpoint Selection Improvement

The current viewpoint selection is performed greedily as explained in Section III-F, however, other methods should be explored taking into account the practicality of the inspection process. For example, it might be best to favour viewpoints far away from the object over closer ones to reduce the risk of collision. Also selection can be performed in conjunction with the path planning to reach a more optimal solution.

D. Viewpoint Filtering Parallelisation

The viewpoint filtering process as defined in Algorithm 2 is currently performed sequentially for each viewpoint. As the filtering process for each viewpoint is independent, it is possible to parallelise the filtering process over multiple processing units. Given this filter is the most computationally intensive components of the algorithm, large gains in total run-time can be realised with this improvement.

VI. Conclusion

The report presents a technical overview of the Coverage Path Planning (CPP) algorithm implemented to perform global trajectory planning for the inspection of a 3D structure such as an aircraft. The report introduces planning concepts, mathematical models for camera based image projection and a pseudo-code outline of the algorithm. Important assumptions on the viewpoint selection and tour planning are highlighted that could be addressed in future studies.

References

[1] E. Meiri, *Perspective Projection OGLdev*, Accessed August 27 2018. [Online]. Available: http://ogldev.atspace.co.uk/www/tutorial12/tutorial12.html.

[2] B. W. Kernighan and S. Lin, *An Efficient Heuristic Procedure for Partitioning Graphs*, 1970. DOI: 10.1002/j.1538-7305.1970.tb01770.x.

[3] Google, *Google Optimization Tools*, Accessed August 10 2018. [Online]. Available: https://developers.google.com/optimization/.

Appendix B

Greedy submodular optimisation proof

A greedy approach for submodular optimisation comes within a factor $1 - e^{-1}$ of the optimal solution. Proof for this is explained in this appendix (Nemhauser *et al.* 1978).

The algorithm starts with empty set M_0 and iteratively adds elements for steps $i = 0, \ldots, (k-1)$, which is formulated as:

$$M_{i+1} = M_i \cup \{\underset{\nu \in V \setminus M_i}{\operatorname{argmax}} f(M_i \cup \{\nu\})\} \tag{B-1}$$

Where ν corresponds to a subset of M_{gp} such that the score should be maximized in order to add it to M_i. The gain is defined by Equation 2-5:

$$\text{gain}(\nu|M) = f(M \cup \nu) - f(M) \tag{B-2}$$

Let $M_i = (\nu_1, \nu_2, \ldots, \nu_i)$ be the observed viewpoints, M^* be the optimal set of viewpoints $M^* = (\nu_1^*, \nu_2^*, \ldots, \nu_k^*)$, $f(\nu)$ a monotone submodular function and OPT $= f(M^*)$ the value of the optimal solution.

Due to the monotone property (adding an element always equals or increases the total score), the following inequality is obtained:

$$f(M^*) \leq f(M^* \cup M_i) \tag{B-3}$$

By using submodularity and with:

$$f(M^* \cup M_i) = f(M_i) + \sum_{j=1}^{k} \text{gain}(\nu_j^* | A_i \cup \{\nu_1^*, \nu_2^* \ldots, \nu_{j-1}^*\}) \tag{B-4}$$

The inequality can be rewritten to:

$$\begin{aligned} f(M^*) &\leq f(M_i) + \sum_{z \in M^*} \text{gain}(\nu_{i+1}|M_i) \\ &\leq f(M_i) + k(f(M_{i+1}) - f(M_i)) \\ \frac{1}{k}(\text{OPT} - f(M_i)) &\leq \text{gain}(\nu_{k+1}|M_i) \end{aligned} \tag{B-5}$$

Now, defining $\delta_i = \text{OPT} - f(M_i)$ implies $\delta_0 = \text{OPT} - f(\emptyset) \leq \text{OPT}$ and

$$\delta_i - \delta_{i+1} = f(M_{i+1} - f(M_i)) = \text{gain}(\nu_{i+1}|M_i) \tag{B-6}$$

And

$$\frac{1}{k}\delta_i \leq \delta_i - \delta_{i+1} \tag{B-7}$$

Then, by recursion and using the bound $1 - x \leq e^{-x}$

$$\begin{aligned} \delta_k &\leq \left(1 - \frac{1}{k}\right)^k \delta_0 \\ &\leq \left(1 - \frac{1}{k}\right)^k \text{OPT} \\ &\leq \frac{1}{e}\text{OPT} \\ \text{OPT} - f(M_k) &\leq \frac{1}{e}\text{OPT} \end{aligned} \tag{B-8}$$

Finally, resulting in the bound:

$$f(M_k) \geq \left(1 - \frac{1}{e}\right) \text{OPT} \tag{B-9}$$

Appendix C

Additional results

This appendix shows additional results that are not treated in Chapter 7. It serves as extra material backing the thesis.

C-1 Cost function evaluation

The same structure of visualisation is considered as explained in the introduction of Section 7-1, keeping one axis constant for the graphs and showing the problem in the top left corner.

For an inspection problem with 32 triangles, corresponding to the first viewpoint of the path inspecting the cube, the optimal position is located at:

$$\mathbf{P}_{opt} = \begin{bmatrix} 0.0 \\ 0.0 \\ 6.2 \end{bmatrix} \tag{C-1}$$

This is illustrated in Figure C-1. Given that the triangles combined form a square, it makes sense the optimal location lies directly above the centre. Moreover, the lower right graph greatly resembles the smaller inspection problem of one triangle as depicted in Figure 7-2.

Figure C-2 illustrates the cost for 32 triangles when an obstacle is present and where the weights are $w_I = 1.0$ and $w_O = 1.0$. The optimal position is located at:

$$\mathbf{P}_{opt} = \begin{bmatrix} -0.85 \\ -0.85 \\ 6.75 \end{bmatrix} \tag{C-2}$$

It can be noted that the lower right image resembles that of Figure C-1. However, there is a 'cloudy' part in the middle, which relates to occlusion of the obstacle. While the true optimum is defined in this graph, optimisation using gradient based methods is prone to local minima. Therefore, the weight of occlusion by obstacles must be raised in order to remove the cloudiness.

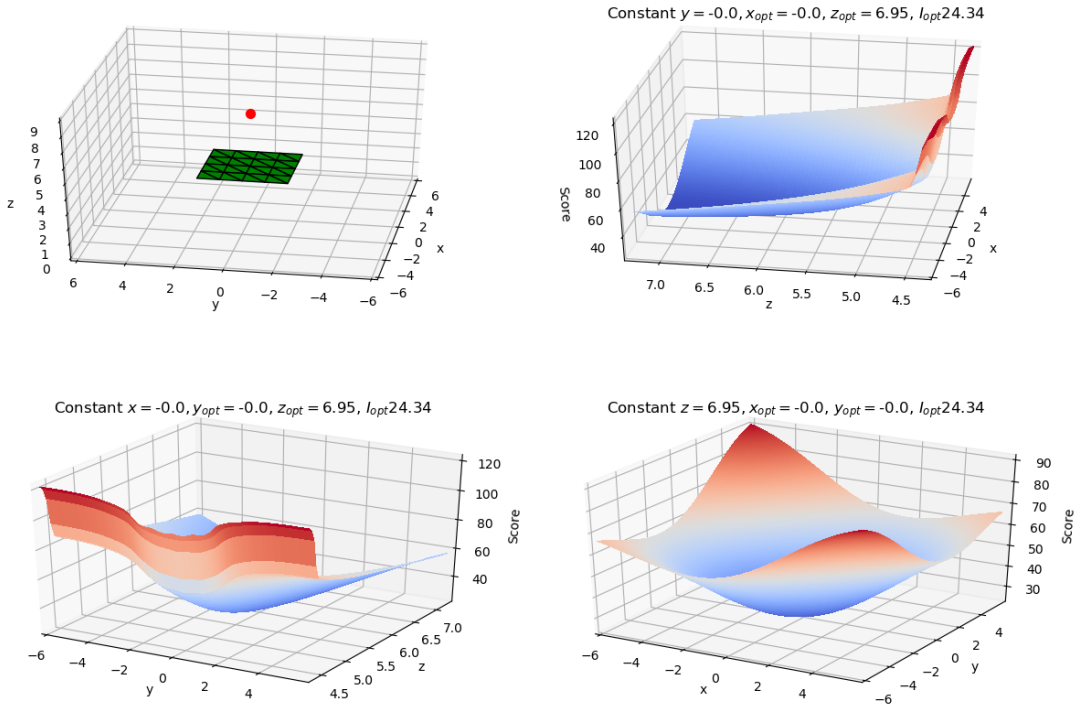


Figure C-1: The score $I(\mathbf{P})$ for all triangles on the optimal position with $w_I = 1.0$ and $w_O = 1.0$.

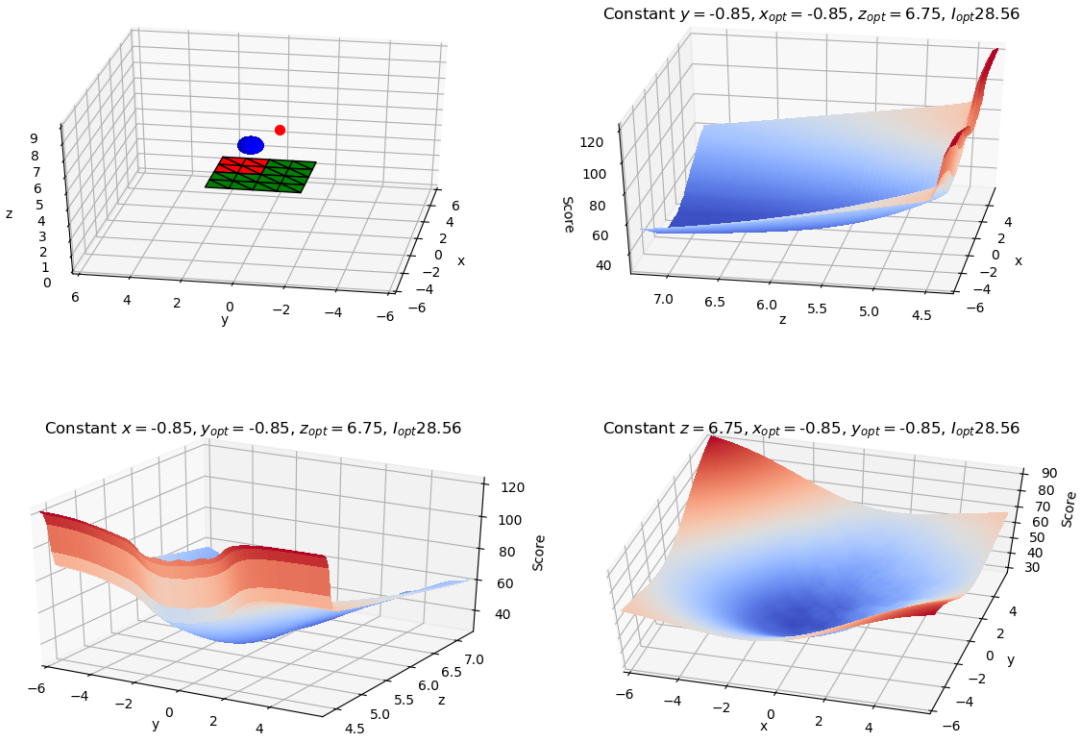


Figure C-2: The score $I(\mathbf{P})$ for all triangles on the optimal position with $w_I = 1.0$ and $w_O = 1.0$.

C-2 Calculation times using Ceres

Ceres is used as the global optimiser of the inspection problems. This section discusses the time it takes Ceres to find a solution. The same problem setting as the previous section is used to collect this data. Due to the structure of the information gain (many sigmoid functions and therefore local minima), the solution is found by adding a small bit of noise and iterating until there is no more change in each axis direction.

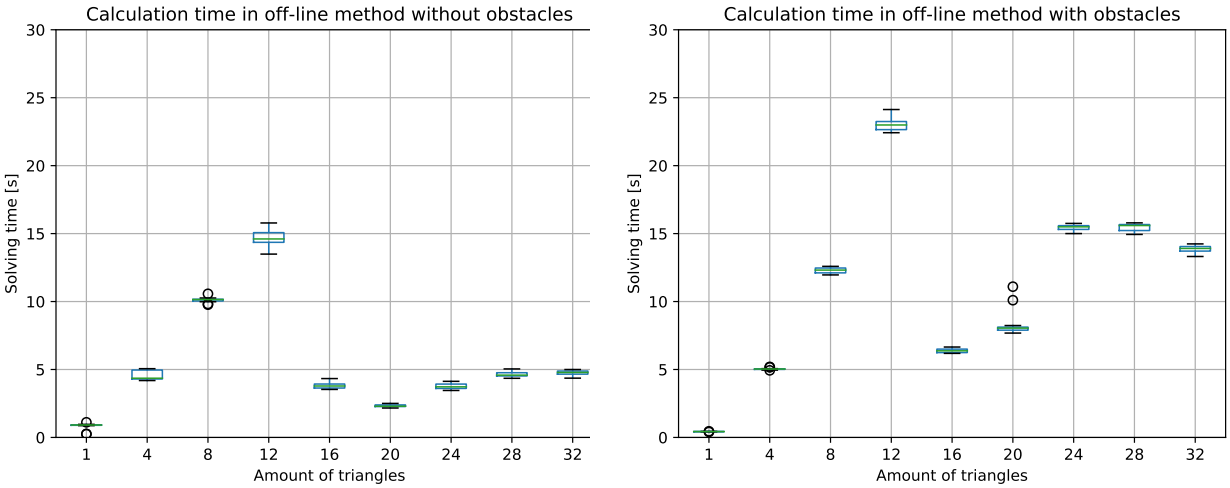


Figure C-3: The calculation time for Ceres to find an optimal solution per size of the problem, expressed in triangles, without obstacles on the left and with obstacles on the right.

The calculation times, without and with obstacles are shown in Figure C-3. What these show, is that the solving times are very problem dependent. After having added 12 triangles, the calculation time seems to drop. This can be explained by the solver jumping to nearby local minima, prolonging the process.

Furthermore, adding obstacles to the problem greatly increases the calculation times. For more than 16 triangles, the duration increases three-fold. This is also line with the complete inspections, shown in Figures 7-7 and 7-8, which show that the off-line now takes much longer to complete than the on-line method.

C-3 Settling times

For point-to-point control, the settling time is also important to consider. Preferably, no excessive overshoot or ringing is desired. Figure C-4 shows the RMSE over time for point-to-point control. From the graph, it can be seen no overshoot is present and no ringing should occur after reaching the target after 11 seconds. The weights have therefore been tuned correctly.

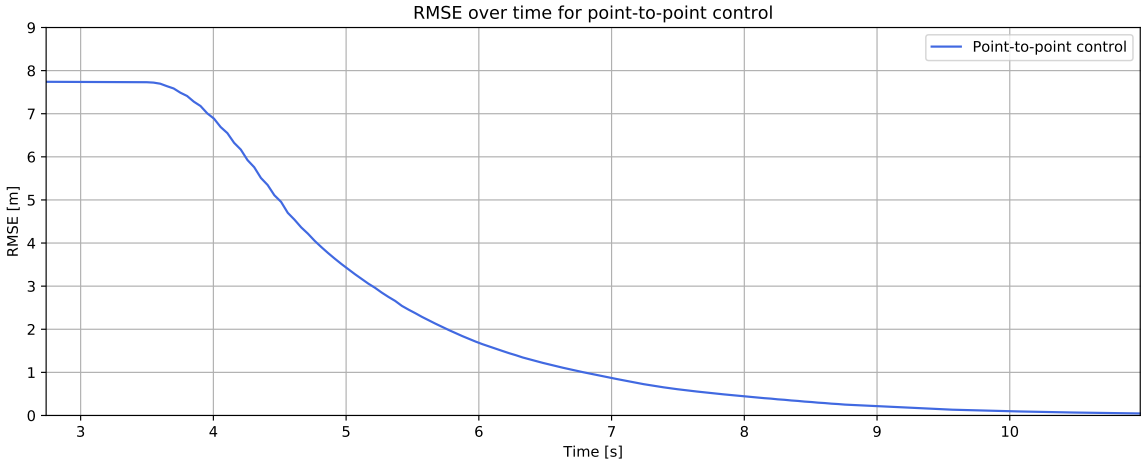


Figure C-4: Settling times point-to-point control.

C-4 Complete inspections

The additional graphs for a complete inspection is the one performed with the global planner method. Data is obtained by processing the same traversed path, where no obstacles are present, in the occluded setting. Figure C-5 shows the minimal obstacle distance and mesh distance, as well as the drone radius that cannot be surpassed. The graph indicates three moments during inspection where collision occurs and one close call.

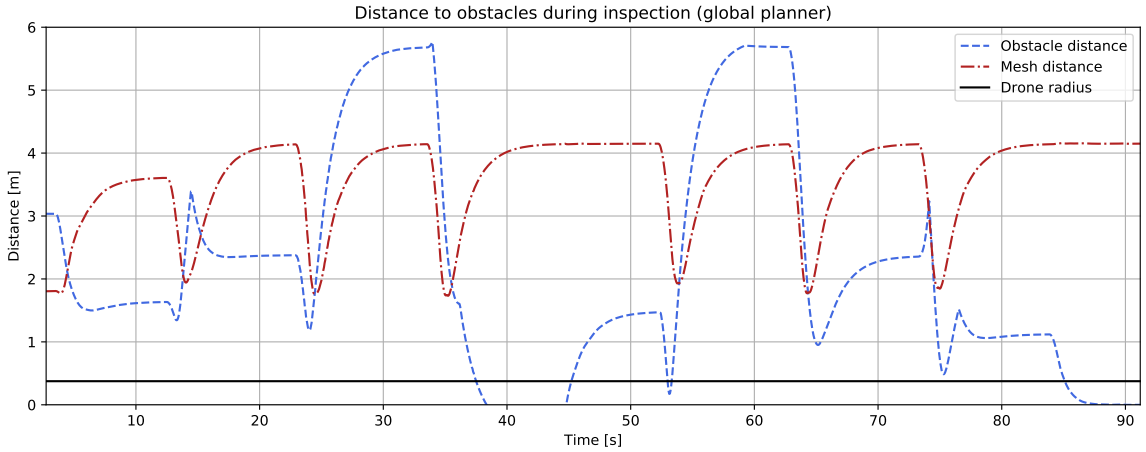


Figure C-5: Distances to obstacles for the P2P method. The distance to an obstacle is defined as the centre of the MAV to the centre of the obstacle.

By comparing Figures 7-10 and 7-11 with Figure C-5, it is clear the cost terms to avoid collision have an effect. However, the desired outcome of no collision is not achieved and more penalty or a different approach should be used to guarantee no collision.

Bibliography

[1] S. Agarwal, K. Mierle, and Others, *Ceres solver*, http://ceres-solver.org.

[2] R. Almadhoun, T. Taha, L. Seneviratne, J. Dias, and G. Cai, "Aircraft Inspection Using Unmanned Aerial Vehicles," *International micro air vehicle competition and conference*, pp. 43–49, 2016. [Online]. Available: http://www.imavs.org/papers/2016/43{_}IMAV2016{_}Proceedings.pdf.

[3] A. Bircher, M. Kamel, K. Alexis, H. Oleynikova, and R. Siegwart, "Receding horizon next-best-view planner for 3D exploration," *Proceedings - IEEE International Conference on Robotics and Automation*, vol. 2016-June, no. May, pp. 1462–1468, 2016.

[4] A. Bircher, M. Kamel, K. Alexis, H. Oleynikova, and R. Siegwart, "Receding horizon path planning for 3D exploration and surface inspection," *Autonomous Robots*, vol. 42, no. 2, pp. 291–306, 2016.

[5] A. Bircher, M. Kamel, K. Alexis, M. Burri, P. Oettershagen, S. Omari, T. Mantel, and R. Siegwart, "Three-dimensional coverage path planning via viewpoint resampling and tour optimization for aerial robots," *Autonomous Robots*, vol. 40, no. 6, pp. 1059–1078, 2016.

[6] S. Bouabdallah and R. Siegwart, "Design and Control of an Indoor Micro Quadrotor," *International Conference on Robotics & Automation*, no. April, pp. 4393–4398, 2004.

[7] DJI, *Dji onboard sdk*, https://developer.dji.com/onboard-api-reference/index.html, 2018.

[8] Y. Emek and A. Rosén, "Semi-Streaming Set Cover," *ACM Trans. Algorithms*, vol. 13, no. 1, 6:1–6:22, 2014. arXiv: 1404.6763. [Online]. Available: http://doi.acm.org/10.1145/2957322.

[9] T. Foote and M. Purvis, *Standard units of measure and coordinate conventions*, https://www.ros.org/reps/rep-0103.html.

[10] G. Frison, *High-performance interior-point-method*, https://github.com/giaf/hpipm.

[11] M. Giftthaler, M. Neunert, M. Stäuble, and J. Buchli, *The Control Toolbox - an open-source C++ library for robotics, optimal and model predictive control*, 2018.

[12] O. Khatib, "Real-Time Obstacle Avoidance For Manipulators and Mobile Robots," *Proceedings. 1985 IEEE International Conference on Robotics and Automation*, vol. 2, pp. 500–505, 1985.

[13] A. Krause and D Golovin, "Submodular Function Maximization," *Tractability: Practical Approaches to Hard Problems*, vol. 94, no. 12, pp. 507–508, 2014.

[14] Y.-Y. Lin, C.-C. Ni, N. Lei, X. David Gu, and J. Gao, "Online Inspection Path Planning for Autonomous 3D Modeling using a Micro-Aerial Vehicle," *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 6217–6224, 2017. [Online]. Available: http://dx.doi.org/10.1109/ICRA.2017.7989583.

[15] Z. C. Marton, R. B. Rusu, and M. Beetz, "On Fast Surface Reconstruction Methods for Large and Noisy Datasets," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, Kobe, Japan, 2009.

[16] D. Q. Mayne, J. B. Rawlings, C. V. Rao, and P. O. Scokaert, "Constrained model predictive control: Stability and optimality," *Automatica*, vol. 36, no. 6, pp. 789–814, 2000.

[17] G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher, "An analysis of approximations for maximizing submodular set functions–i," *Math. Program.*, vol. 14, no. 1, pp. 265–294, Dec. 1978. [Online]. Available: https://doi.org/10.1007/BF01588971.

[18] N. Potdar, "Airviewtour : Viewpoint Selection and Tour Planning," 2018.

[19] M. Quigley, B. Gerkey, K. Conley, J. Faust, T. Foote, J. Leibs, E. Berger, R. Wheeler, and A. Ng, "Ros: An open-source robot operating system," in *Proc. of the IEEE Intl. Conf. on Robotics and Automation (ICRA) Workshop on Open Source Robotics*, Kobe, Japan, May 2009.

[20] K. Ring, *Horizon culling*, https://cesium.com/blog/2013/04/25/horizon-culling/.

[21] M. Roberts, S. Shah, D. Dey, A. Truong, S. Sinha, A. Kapoor, P. Hanrahan, and N. Joshi, "Submodular Trajectory Optimization for Aerial 3D Scanning," *Proceedings of the IEEE International Conference on Computer Vision*, vol. 2017-Octob, pp. 5334–5343, 2017. arXiv: 1705.00703.

Glossary

List of Acronyms

3mE	Mechanical, Maritime and Materials Engineering
DCSC	Delft Center for Systems and Control
MAV	Micro Aerial Vehicle
MAVs	Micro Aerial Vehicles
MBI	Mainblades Inpections
LIDAR	Laser Imaging Detection and Ranging
RRT	Rapidly exploring Random Tree
MPC	Model Predictive Controller
CPP	Coverage Path Planning
BVS	Boundary Value Solver
OP	Orienteering Problem
SOP	Submodular Orienteering Problem
IPP	Informative Path Planning
NBV	Next-best-view
TSP	Travelling Salesman Problem
ISAAC	Intelligent Sensing and Automated Control
FOV	Field of View
ROS	Robotic Operating System
RRT	Rapidly-exploring Random Trees
RMSE	Root mean squared error
P2P	Point-to-Point
HPIPM	High-performance interior-point-method

List of Symbols

Greek symbols

ν	$\mathbb{R}^6$	Viewpoint vector
δ	[rad]	Allowed maximum incidence angle
η		Slope for sigmoid function
γ	[rad]	Incidence angle
ψ	[rad]	Pitch angle
θ	[rad]	Roll angle
φ	[rad]	Yaw angle
ξ		Inspection path, consisting of ordered viewpoints

Roman symbols

$\mathbf{T}_i$	$\mathbb{R}^3$	Centre vector of ith triangle
$\mathbf{A}$	$\mathbb{R}^3$	Vertex lying of a triangle
$\mathbf{B}$	$\mathbb{R}^3$	Vertex lying of a triangle
$\mathbf{C}$	$\mathbb{R}^3$	Vertex lying of a triangle
$\mathbf{n}_i$	$\mathbb{R}^3$	Normal of ith triangle
$\mathbf{O}_c$	$\mathbb{R}^3$	Centre of obstacle
$\mathbf{r}_O$	$\mathbb{R}^3$	Radii of obstacle
$\mathbf{t}$	$\mathbb{R}^3$	A point lying on triangle T
$\mathbf{P}$	$\mathbb{R}^3$	Camera position
$\mathbf{x}$	$\mathbb{R}^6$	State vector
$\mathbf{u}$	$\mathbb{R}^6$	Input vector
$\mathbf{n}_{i,A,AB^-}$	$\mathbb{R}^3$	Constraint normal vector originating from point **A** by rotating around edge **AB**
$\mathbf{h}_c$	$\mathbb{R}^3$	Hook vector to constraint plane
a	[m]	Activation distance for penalty
A	[m^2]	Surface area of object to inspect
A_I	[m^2]	Covered surface area of all images
A_{gp}	[m^2]	Covered surface area by the global planner algorithm
B		Object to inspect

c		Constant for deriving optimal slope sigmoid functions
d_{max}	[m]	Allowed maximum distance
d_{min}	[m]	Required minimum distance
F_{thrust}	[N]	Force by all rotors
I		Image taken at a viewpoint
J		Cost function
k	[m]	Projection distance for sigmoid function
m	[kg]	Mass of MAV
M		The mesh model of the object to inspect
M_{gp}		The mesh inspected by the global planner algorithm
m_k		Set of triangles viewed from kth viewpoint
$m_{infeasible}$		Set of infeasible triangles during inspection
$m_{inspected}$		Set of triangles viewed during inspection
n		Amount of triangles in the mesh
p		Amount of viewpoints in $\mathcal{V}$
Q		Weighting matrix for states
r_{drone}	[m]	Radius of bounding box of MAV
T_i		ith triangle in the mesh
$\mathcal{O}$		Set of obstacles
$\mathcal{V}$		Set of viewpoints
$\mathcal{W}$	$\mathbb{R}^3$	Environment in which the problem resides
w_{height}		Weight for height
$w_{incidence}$		Weight for incidence angle
w_I		Weight for information gain
w_O		Weight for obstacle distance penalty
w_M		Weight for mesh distance penalty
R		Weighting matrix for input
x	[m]	Position along x-axis
y	[m]	Position along y-axis
z	[m]	Position along z-axis
$\emptyset$		Empty set