Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Engineering Cybernetics

Åshild Berg Rosland

Robot Based Inspection of Valves in Industrial Facilities

Master's thesis in Cybernetics and Robotics
Supervisor: Morten Omholt Alver
Co-supervisor: Jonas Peter Sørensen, Øystein Barth Utbjoe
June 2023

NTNU
Norwegian University of Science and Technology

Åshild Berg Rosland

Robot Based Inspection of Valves in Industrial Facilities

Master's thesis in Cybernetics and Robotics
Supervisor: Morten Omholt Alver
Co-supervisor: Jonas Peter Sørensen, Øystein Barth Utbjoe
June 2023

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Engineering Cybernetics

Robot Based Inspection of Valves in Industrial Facilities

Author:
Åshild Berg Rosland

Supervisor:
Morten Omholt Alver

Co-supervisors:
Jonas Peter Sørensen
Øystein Barth Utbjoe

Master's thesis
Department of Engineering Cybernetics
Norwegian University of Science and Technology

June 1, 2023

Preface

The research presented in this thesis was conducted as part of my Master's degree program in Cybernetics and Robotics at NTNU in collaboration with Equinor. The thesis is a continuation of the work conducted in the specialization project [1], with the purpose of improving an algorithm's efficacy and scalability for finding optimal inspection positions for a robot to inspect valves in industrial facilities. This work was carried out over a period of 21 weeks.

Acknowledgements

I would like to express my gratitude to my supervisor, Morten Omholt Alver, for his guidance and support throughout the research process. His valuable insights and feedback were instrumental in shaping the study and improving the quality of the research. I would also like to express my appreciation to my co-supervisors at Equinor, Jonas Peter Sørensen and Øystein Barth Utbjoe, for their assistance and contributions to this project. Their expertise within the oil and gas industry and technical guidance throughout the project greatly facilitated the progress of my work.

Executive summary

This study aims to develop an algorithm for computing optimal inspection positions for valves in industrial facilities. The thesis builds upon the specialization project [1] and addresses several sub-objectives. These include integrating the planning algorithm into Equinor's system, adapting the optimization algorithm to ensure scalability for larger facilities, considering movable camera positions for optimizing inspections, and providing recommendations for future research in this domain.

The algorithm's optimization process is designed as a search algorithm, incorporating factors such as the distance between the target valve and the walkway, the angle between the camera direction and the optimal inspection direction of the valve, and the camera's clear view of the target valve. Additionally, the algorithm accounts for robot characteristics in the vertical direction by defining a height interval for each potential inspection position.

The results demonstrate that the algorithm significantly reduces the run time but compromises the accuracy of finding suitable inspection points. The search technique employed is a local optimization approach that exploits the minimum distance between the target valve and the walkway for defining the search space, potentially overlooking better inspection positions that may have slightly longer distances. Furthermore, finding the next search point relies heavily on the orientation of the target valve, posing challenges for valves that are not aligned with the ground floor. Additionally, incorporating robot characteristics solely in the vertical direction proves unnecessary without considering horizontal characteristics for valves that are difficult to view from the walkway, and reduces the accuracy further.

Based on these findings, the thesis proposes further work, suggesting the implementation of a repetitive search to avoid restricting the algorithm to a limited region based solely on the minimum distance from the target valve to the walkway line. It also recommends incorporating robot characteristics in the horizontal plane to enable inspections of valves not fully visible from the walkway.

Overall, this thesis establishes the feasibility of robot based inspection of valves and sensors in industrial facilities, offering valuable insights for future real-life applications. Further research in this area holds the potential to enhance worker safety, reduce environmental impact, and lower maintenance costs in industrial facilities.

Sammendrag

Formålet med denne masteroppgaven er å utvikle en algoritme for å beregne optimale inspeksjonsposisjoner for ventiler i industrianlegg, basert på funn fra fordypningsprosjektet [1]. For å oppnå dette er det utarbeidet flere delmål, inkludert integrasjon av planleggingsalgoritmen i Equinors system, tilpasning av optimaliseringsalgoritmen for å sikre skalerbarhet for større anlegg, vurdere bevegelige kameraposisjoner for optimalisering av inspeksjoner, og gi anbefalinger for fremtidig arbeid.

Algoritmens optimeringsprosess er utformet som en søkealgoritme, som tar hensyn til tre faktorer: avstanden mellom målventilen og gangveien, vinkelen mellom kameraretningen og den optimale inspeksjonsretningen til ventilen, og kameraets klare sikt mot målventilen. I tillegg tar algoritmen hensyn til robotkarakteristikk i vertikal retning ved å definere et høydeintervall for hver potensielle inspeksjonsposisjon.

Resultatene viser at algoritmen reduserer kjøretiden betydelig, men på bekostning av nøyaktigheten til å finne passende inspeksjonspunkter. Søketeknikken i bruk er en lokal optimaliseringstilnærming, som alltid starter søket basert på minimumsavstanden mellom målventilen og gangveien, og dermed avgrenser søkeområdet. Dette fører til at algoritmen overser inspeksjonsposisjoner som kan ha litt lengre avstander, men som potensielt er bedre egnet for inspeksjon. I tillegg avhenger iterasjonene til neste søkepunkt sterkt av orienteringen til målventilen, som gir utfordringer for ventiler som ikke har horisontal orientering. Resultatene viser også at inkludering av robot karakteristikker kun i vertikal retning ikke bedrer algoritmens ytelse i å finne gode inspeksjonsposisjoner for ventiler som er utfordrende å inspisere fra gangveien, og reduserer antall gode inspeksjonsposisjoner.

Basert på disse funnene gir studiet forslag til videre arbeid, deriblant implementering av et repeterende søk for å unngå å begrense algoritmen til et område basert utelukkende på minimumsavstanden fra målventilen til gangveilinjen. Den anbefaler også å inkludere robotkarakteristikk i horisontalplanet for å muliggjøre inspeksjoner av ventiler som ikke er fullt synlige fra gangveien.

Oppgaven etablerer at det er gjennomførbart å utvikle algoritmer for robotbasert inspeksjon av ventiler og sensorer i industrielle anlegg, og gir verdifull innsikt for fremtidige applikasjoner i virkelige anlegg. Ytterligere forskning på dette området har potensialet til å øke arbeidernes sikkerhet, redusere miljøpåvirkningen og redusere vedlikeholdskostnadene på industrianlegg.

Table of Contents

List of Tables

List of Figures

List of Source Codes

Abbreviations

Abbreviation	Description
Industry 4.0	The fourth industrial revolution
IoT	Internet of Things
DT	Digital Twin
CBM	Condition Based Maintenance
SLAM	Simultaneous Localization and Mapping
ISAR	Integration and Supervisory Control of Autonomous Robots
DFO	Derivative Free Optimization

1 Introduction

The fourth industrial revolution, referred to as Industry 4.0, aims to integrate smart and autonomous technologies into various industrial sectors [2], including the oil and gas industry. The integration of technologies such as the Internet of Things (IoT), artificial intelligence, automation and digital twins (DTs) has the potential to increase efficiency, productivity, and worker safety in manufacturing processes.

Oil and gas operations are highly complex and conducted in challenging environments [3]. The various components of these facilities are prone to failures and faults, causing the components to require maintenance to prevent accidents and protect workers, the environment, and operational costs. To minimize these risks, predictive measures such as Condition Based Maintenance (CBM) are often implemented, which measures various states in components and enables the possibility of detecting changes in the measured states. Based on changes in measurements one may predict when a failure may occur, and be able to plan and schedule corrective maintenance to hinder accidents, environmental contamination or financial losses by addressing the problems before they become critical.

A disadvantage of CBM is that it relies solely on sensor data, which are themselves prone to faults. This can lead to inaccurate or wrong measurements, and prevent a complete evaluation of the situation. Therefore, visual inspections are still considered a crucial part of predictive maintenance, as it provides a comprehensive understanding of the situation and may help discover and address potential issues that are not described completely by the CBM.

1.1 Background

1.1.1 Motivation

The utilization of valves in the oil and gas industry is crucial due to the widespread utilization of pipeline systems in production sites. These valves serve to regulate the flow of substances within pipes, including the speed, temperature, and pressure of the flow [4, p. 2]. Within a single facility, it is not uncommon to find hundreds or even thousands of valves, each of which necessitates routine maintenance to mitigate the potential for faults and failures. Traditionally, human-performed inspections are conducted at predetermined intervals, as the transportation of personnel to these industrial sites is time-consuming and costly. However, it is possible for failures to occur outside of these inspection windows, necessitating immediate action and limiting the effectiveness of cost-efficient maintenance procedures.

Furthermore, the failure of equipment poses significant risks to the safety of workers, as the machinery involved in oil and gas production can be complex and potentially hazardous. The risk of ignition of hydrocarbons, which may cause fires or explosions, is particularly concerning in the oil and gas industry [5]. To mitigate these risks and improve overall health, safety, and environmental conditions within the workplace, the thesis proposes the implementation of robots for inspection purposes. This approach aims to detect faults early and facilitate preventative maintenance, reducing downtime and improving the overall safety of the plant.

1.1.2 Related research

The current project is an extension of the previous specialization project [1] which was based on a Master's thesis written by E. Blomseth [6]. The earlier work involved developing an algorithm to identify optimal inspection positions for performing inspections on valves based on a digital twin of the industrial facility. An inspection position was considered optimal if it could capture all necessary information required for inspections, and was located in a position in front of the valve with restrictions on the robot's movement. The algorithm calculated the optimal inspection points for each valve by evaluating all possible inspection points along the walkway in the facility. Each point was scored based on three factors: the distance between the camera and the target object, the angle between the camera direction and the optimal direction for inspecting the target object, and the presence of any obstructions blocking the camera's view toward the target valve. The scores were then combined in a scoring function that weighed each factor according to its relative importance, with the inspection position with the lowest score chosen as optimal. Specifically, a position combined of low distance between the camera and the target valve, a small angle between the camera direction and the optimal direction for inspecting the valve, and a clear view towards the target valve would be considered optimal.

Although the algorithm developed in [1] found suitable inspection positions for most valves, there are several issues that still need to be addressed. The algorithm is based on several assumptions and simplifications that do not hold in real-life scenarios. The algorithm assumes that the robot can only move along the middle of the walkway, limiting the possibility of identifying inspection positions for different robots that may not be at the center of the walkway, possibly ignoring a valid inspection point contained within the walkway that may produce the real optimal inspection position on the walkway. Additionally, the algorithm views the camera and robot as a single entity, which does not account for differences in size and camera placement among different robots. This implies that the camera must be positioned on the walkway for inspections, which makes it difficult, or even impossible in some cases, to locate optimal inspection points for valves that are not visible from the walkway.

Furthermore, the algorithm is not efficient in identifying suitable inspection positions.

The algorithm evaluates every possible inspection point sampled along the walkway line for each target valve using complex ray-tracing schemes, which can be time-consuming and computationally expensive. To make the algorithm scalable for use in industrial settings, it is necessary to optimize the run time of the algorithm. Developing a scheme that limits the evaluation of uninteresting inspection points is crucial for real-life applications of the algorithm.

1.2 Problem description

Equinor's core objective is to create and deliver energy in accordance with their three strategic pillars: Always safe, high value and low carbon [7]. Effective maintenance routines are essential for ensuring safe operating conditions for both humans and the environment, while also providing cost savings through minimized plant downtime. As discussed in Section 1.1, one approach to achieving these goals is to implement robotic inspections in industrial facilities.

The primary objective of this project is to develop an algorithm for robotic inspection of industrial facilities to enhance safety, reduce environmental impact, and increase cost savings. Since the existing algorithm for robotic inspections proposed in [1] has limitations in determining the optimal inspection position, the Master's project aims to overcome these limitations by improving the algorithm's efficiency to make it suitable for large facilities and by incorporating the robot's characteristics into the algorithm to ensure the assessment of appropriate points.

The achievement of the primary objective is reliant on five sub-objectives that have been identified for implementation in the project:

1. Integration of the planning algorithm into Equinor's system and the execution of test inspections using a simulated robot, and if feasible, a physical robot.

2. Adaptation of the optimization algorithm to enable scalability with respect to run time for larger facilities.

3. Introduction of instructions for the placement of the camera in relation to the robot's position, both vertically and potentially horizontally.

4. Consideration of a movable camera position in optimizing the inspection position.

5. Provision of recommendations for future research.

Overall, these sub-objectives represent critical steps toward achieving the primary objective and are essential for the successful completion of the project.

1.3 Research approach

In this project a systematic review of literature about previous specialization projects [1][8] and E. Blomseth's Master's thesis [6] was initially conducted. This was followed by a comprehensive exploration of the oil and gas industry and an investigation of relevant theories that could apply to robot based inspections for industrial facilities. Throughout the project, a series of meetings were held with the supervisor and co-supervisors to discuss potential solutions and ideas and to obtain insights into Equinor's current solutions and how the thesis could contribute to the industry.

1.4 Delimitations

The aim of the developed algorithm is limited to deciding the correct location for performing inspections and does not consider navigation to the inspection points of choice. The algorithm relies exclusively on the digital twin of the Huldra facility, thereby limiting the scope of testing to the Huldra platform. Physical testing on the plant was deemed unfeasible, with only Gazebo simulations and Equinor's simulation tool serving as viable substitutes. Additionally, the algorithm focuses solely on valves and has yet to be tested with other industrial equipment or sensors.

1.5 Structure of the report

The thesis is divided into six chapters. Chapter one introduces the research topic, including background information and the study's objective. Chapter two describes the theoretical concepts used in the study. Chapter three presents the methods used in the study, including data collection and analysis procedures. Chapter four presents the results of the study. Chapter five discusses the implications of the study's results, and suggests directions for future research in this area. Finally, chapter six summarizes and concludes the findings of the project.

2 Theory

This chapter provides a theoretical framework for understanding the key concepts, principles, and models underpinning the study. This is accomplished by providing a clear definition of technical terms to ensure that the reader understands the terminology used throughout the report, as well as describing how the concepts are relevant to the problem at hand.

2.1 Digital twins

A digital twin is a digital representation of a physical entity or system [9]. It contains virtual information about the physical system and may be used to collect data from the physical system in order to perform tests or simulations. Moreover, a digital twin has the capacity to describe the dynamics and states of the physical system in the digital world, enabling convergence between physical and virtual states [10, p. 1]. Additionally, it can utilize measurements obtained from the physical entity to synchronize with the actual states.

The following paragraph is taken from the specialization project [1, p. 6]. The project uses Equinor's digital twin of the Huldra platform, allowing extraction of a 3D model of the industrial facility to be used for testing the developed algorithm in a simulated environment. As a digital twin contains information about all physical objects within the model, the design in the model is expected to be a replica of the actual real world. However, the digital twin of the Huldra model used in the project does not account for changes in the environment. This means that although the physical objects' representation is accurate, the state of the objects may not be represented through the digital twin. This includes the presence of paint breaches, spills or humans in the facility, which may have a negative impact on the quality of inspections as the robot may not be able to move to its optimal position. The digital twin can be viewed as a ”perfect” version of the real world, as it allows for the simulation and optimization of the inspection process without the interference of these factors.

2.2 3D models

This section is taken from the specialization project [1, p. 6].

A 3D model is a digital representation of surfaces of one or multiple objects in 3D-space [11]. There are different types of representations of the 3D models. A common one, which is used in this project, is to describe each surface by a set of vertices [12]. Each set of vertices is then stored within a so-called *face*, which contains the indices of each of the corresponding vertices that construct a particular surface.

3D models can be stored in several file formats for viewing, importing and exporting 3D files [12]. This project uses the file format *obj*. It is a text-based file format, which may specify different properties of the 3D object, including vertices, faces, lines and vertex normals. Each line in the file starts with a key, describing which type of information the line contains.

An example of the *obj* file from the 3D representation of the Huldra platform is given in Listing 2.1. Each line starts with a key, in this case either "o", "v", "vn" or "f". A line starting with the letter "o" specifies which object the following commands describes. The key "v" indicates that the line expresses a set of coordinates (x, y, z) of a vertex, while "vn" gives the vertex normals of the corresponding vertex indices. The key "f" describes a face, which is in this case constructed by three vertices, resulting in a triangular face. Each face is described by two numbers on the form "v//vn", where the first number is the vertex index and the second number is the corresponding normal index.

Listing 2.1: Excerpt of obj-file from the Huldra platform.

```
o BOX_6_of_SUBSTRUCTURE_/ESCAPE-MEZZ/FLOOR_Mesh

v -109.999969 29.460001 310.825012
v -109.999969 29.440001 310.825012
v -109.999969 29.460001 311.825012
...
vn 1.0000 0.0000 0.0000
vn -1.0000 0.0000 0.0000
vn 0.0000 -1.0000 0.0000
...
f 3//1 2//1 1//1
f 3//1 4//1 2//1
f 7//2 6//2 5//2
...
```

The descriptions given by the object files can be exploited by software to generate visualizations of the 3D model. For visualizations in this project, the Python library *TriMesh* is utilized [13], which enables reading, writing, manipulation and analysis of 3D meshes.

The 3D model of the Huldra platform is visualized using the Gazebo software tool.

2.3 Optimization

The following two subsections are taken from the specialization project [1, p. 7].

Optimization is the process of finding the optimal solution to an objective function, which quantifies the definition of optimal by some metric [14, p. 2]. The objective function consists of variables desired to be chosen such that the objective function is optimized, often given some constraints on the variables that must be satisfied in order for the solution to be valid. This optimization could mean either minimizing or maximizing the objective function with respect to the unknown variables, based on the application specific problem at hand.

Optimization is an important tool exploited in the project, as the purpose is to find optimal inspection positions for the robot given constraints on where the robot is allowed to be located. No numerical optimization techniques will be used, but an objective function to be minimized based on some characteristics of the possible positions will be investigated [6, p. 5].

2.3.1 Weighting of objective functions

The objective function to be minimized may consist of several parameters. The objective function can then be formulated as

$$Minimize_{\mathbf{x}}\mathbf{f}(\mathbf{x}) = [f_1(\mathbf{x}), f_2(\mathbf{x}), ..., f_N(\mathbf{x})]$$

where **x** are the decision variables to be minimized, and $\mathbf{f}(\mathbf{x})$ are the objective functions.

Each parameter may not be equally critical in reaching the desired state, introducing the need of *weighting* the objectives in the objective function [15, p. 1]. Objectives may conflict with each other, causing the solution of a weighted minimization problem to not necessarily have the optimal value for each parameter isolated. The solution is to use the concept of weighting, which indicates which objectives that are preferred to be minimized. This is often done by introducing a scaling function of the weighted sum of the parameters, with weighting coefficients that represent the importance of each objective relative to the other objectives. In mathematical terms, this can be formulated as

$$\phi(f_1(\mathbf{x}), ..., f_N(\mathbf{x})) = \sum_{i=1}^{N} w_i f_i(\mathbf{x})$$

where ϕ represents the function of the weighted sum of the different objective functions.

2.3.2 Optimization using search algorithms

In the domain of problem-solving, optimization is accomplished by means of optimization algorithms that rely on an iterative process, beginning with an initial estimate and generating subsequent estimates until a solution is reached [14, p. 8]. The task of discovering an optimal solution from a domain of potential solutions is known as a search problem [16, p. 83]. Typically, such problems are comprised of an initial state, a set of feasible actions, and a specified goal state that the system aims to achieve. To address this challenge, a search algorithm navigates through the search space by exploring potential solutions and evaluating them according to a predetermined set of criteria, with the goal of selecting the most favorable solution [16, p. 89].

To illustrate how such a search algorithm may be constructed, the following subsections will outline briefly how two specific search algorithms work. Both methods aim to find the minimum of the given objective function.

Line search methods: Steepest descent

A line search method computes for each iterate the search direction p_k, and then decides how far to move along this direction [14, p. 30]. Each iteration is given by

$$x_{k+1} = x_k + \alpha_k p_k \tag{2.1}$$

where x_{k+1} is the next iterative point, x_k is the current iterative point, α is the step length and p_k is the search direction. Each line search method computes the search direction in different ways.

The most basic line search method is the steepest descent method [14, p. 21]. It chooses the search direction based on the direction of the objective function, f, that decreases the most rapidly. The algorithm finds this direction by exploiting the gradient vector of the objective function at that point, ∇f_k. The search direction found by the steepest descent method is thus given by

$$p_k = -\nabla f_k \tag{2.2}$$

where ∇ is the gradient of the objective function.

Figure 2.1 shows a graph where $y = f(x)$ is a quadratic function of x. The red circles illustrate each iteration of the point found by the steepest descent method, where at each iteration, the point converges towards the minimum of the function $f(x)$ by following the steepest direction of the function.

Derivative free optimization methods: Nelder-Mead

The derivatives of the optimization functions are not always available for use, causing the need for derivative free optimization techniques [14, p. 220]. Derivative Free Optimization (DFO) methods use the function values at a set of sample points to determine a new iterate by some other means.

One DFO method is the Nelder-Mead simplex algorithm [14, p. 238], which exploits a geometric approach to explore the search space. The algorithm forms a simplex, i.e. a geometrical shape that depends on the dimensionality of the problem. For a problem of N variables, the formed simplex will have $N+1$ number of vertices. For each iteration of the algorithm, the simplex is updated by evaluating the objective function for each vertex of the simplex, and ordering the vertices from the highest to lowest objective function value (i.e. worst to best). Each iteration seeks to remove the vertex with the worst function value and replace it with another point yielding a better function value. This is done by computing the centroid of all vertices in the simplex, excluding the worst vertex, and performing a transformation of the worst vertex point.

An example of how such a transformation may proceed is shown in Figure 2.2. The black simplex forms the base simplex, and consists of three vertices: v_0, v_1, v_2. Each vertex is ordered such that the vertex with the highest subscript is the vertex giving the highest objective function value. In this case, this is the vertex v_2, which is to be removed

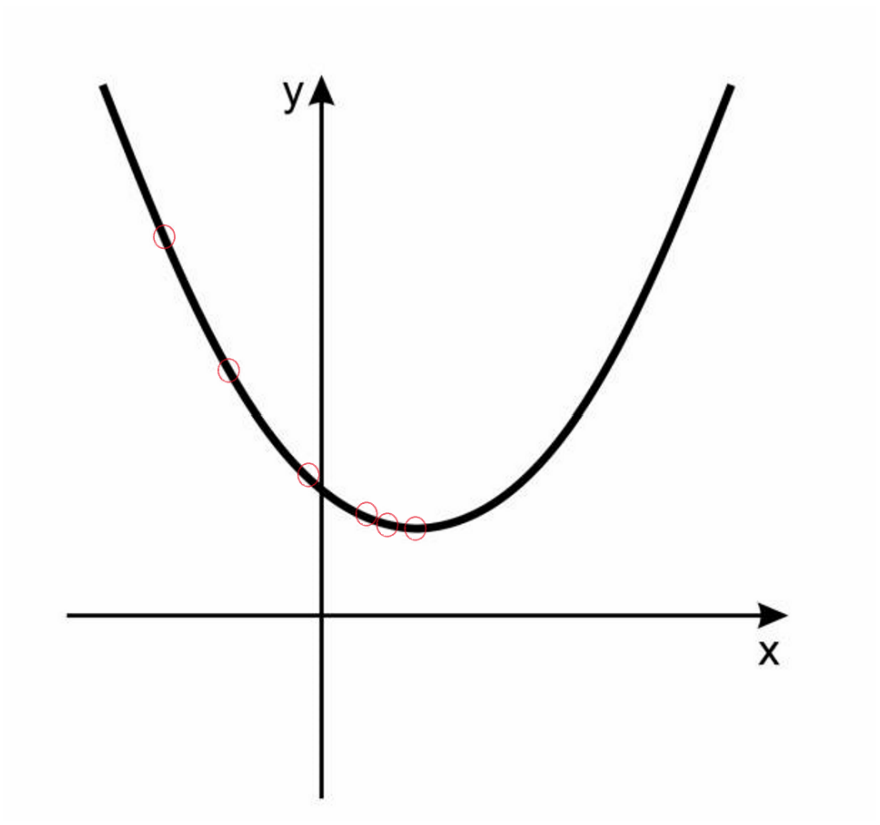


Figure 2.1: Iterations of the steepest descent method. The red circles illustrate the iterative points found by the steepest descent method. It converges towards the minimum of the function $y = f(x)$.

by the algorithm. It is done by so-called *reflection* of the point over the centroid, v_c, of the vertices v_0 and v_1, resulting in the point called v_r. The new point v_r is evaluated in the objective function, and based on the result it is decided to move the point in the direction of the two other vertices, or if other vertices are to be moved closer to v_r if it results in the lowest function value among the other vertices of the simplex. Iterations of this sort are done until the solution converges, or until it meets a given termination criteria.

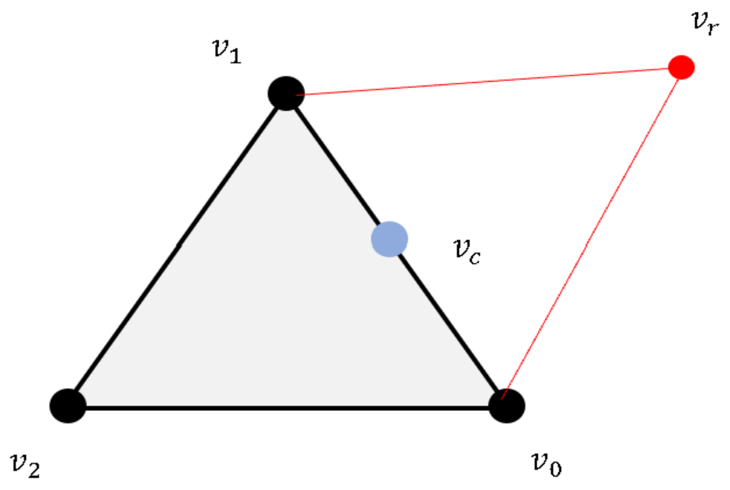


Figure 2.2: An iteration of the Nelder Mead algorithm.

2.4 Quaternions

Quaternions are one way to describe orientations in 3D space by using an ordered set of four numbers; a scalar q_0 and a vector $\mathbf{q} = (q_1, q_2, q_3)$. This can be formulated as $q = q_0 + q_1\hat{\mathbf{i}} + q_2\hat{\mathbf{j}} + q_3\hat{\mathbf{k}}$, where q_0, q_1, q_2, q_3 are real numbers and $\hat{\mathbf{i}}, \hat{\mathbf{j}}, \hat{\mathbf{k}}$ are quaternions units [17, p. 8949]. This may be reformulated into a compact form of $\mathbf{q} = (q_0, q_1, q_2, q_3)$. The vector part of the quaternion represents the axis about which a rotation will occur, whereas the scalar component of the quaternion represents the amount of rotation that will occur about this axis. Thus, quaternions are used as an operator to stretch and rotate a vector around a specified axis. The rotation of a point using quaternions can be described by performing three steps [18]:

1. Convert the point to be rotated into a quaternion by setting the coordinates of the point as the vector components of the quaternion, and setting the scalar factor of the component to zero. Mathematically, the point $\mathbf{p} = (p_1, p_2, p_3)$ to be rotated is rewritten as a quaternion on the form $\mathbf{q} = (0, p_1, p_2, p_3)$.

2. The rotation of the point is done by performing a quaternion multiplication: the

point to be rotated is pre-multiplied with the rotation quaternion and post-multiplied with the conjugate of the rotation quaternion. Mathematically, the multiplication that is performed is given by

$$\mathbf{p}' = \mathbf{q}\mathbf{p}conj(\mathbf{q})$$

where $\mathbf{p}'$ contains the coordinates of the rotated point, $\mathbf{q}$ is the rotation quaternion, $\mathbf{p}$ is the point to be rotated, and $conj()$ represents the conjugate function.

3. Extract the rotated coordinates from $\mathbf{p}'$, i.e. the vector part of the quaternion $\mathbf{p}'$.

2.5 SLAM

Simultaneous Localization and Mapping (SLAM) is a problem where an agent (i.e. robot) simultaneously creates a map of the surrounding unknown environment while detecting where it is within the map [19]. The agent is thus affected by uncertainties in both the localization and mapping stage of the technique due to inaccurate measurements of sensors and therefore exploits probabilistic tools in solving the SLAM problem [20, p. 122].

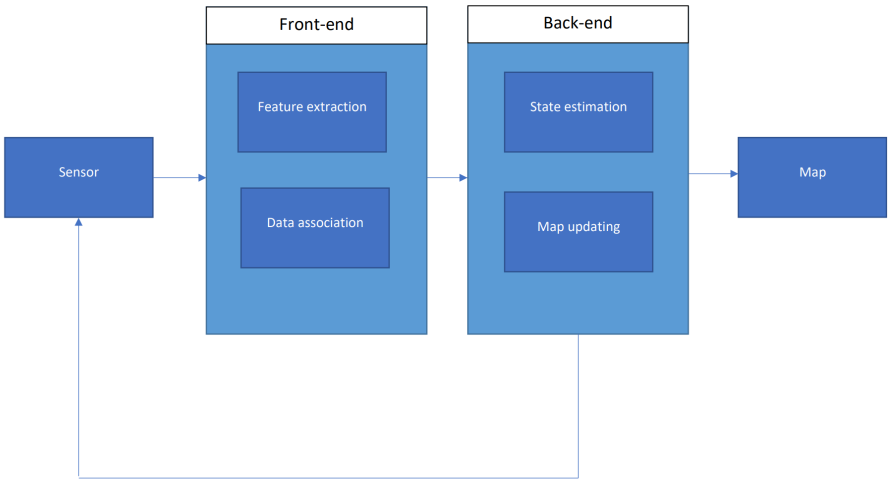


Figure 2.3: Overview of the SLAM approach.

Figure 2.3 illustrates the main building blocks of SLAM, involving sensors, front-end data analysis and back-end estimations to build the final mapping of the environment [20, p. 121]. The process may be summarized by five steps:

1. **Sensing**: The use of sensors such as LIDAR scanning, odometer or cameras in order to collect information about the environment.
2. **Feature extraction**: Extraction of specific features available in the sensor data, i.e. corners, edges or landmarks.

3. **Data association**: Association of extracted features with features from past sensor measurements to establish correspondences.

4. **State estimation**: Estimation of the agent's position and orientation (i.e. pose) based on the established correspondences.

5. **Map building**: Updating the map based on the pose estimations and associated features.

These steps are iteratively performed and the state estimation and map building are refined with each new measurement captured from the sensors, until convergence and thereby reaching a final mapping of the environment.

2.6 TurtleBot

TurtleBot is a ROS standard platform robot [21], being both a physical and simulated robot. There are multiple versions of TurtleBot, but this project will use the TurtleBot3 Waffle model, depicted in Figure 2.4.

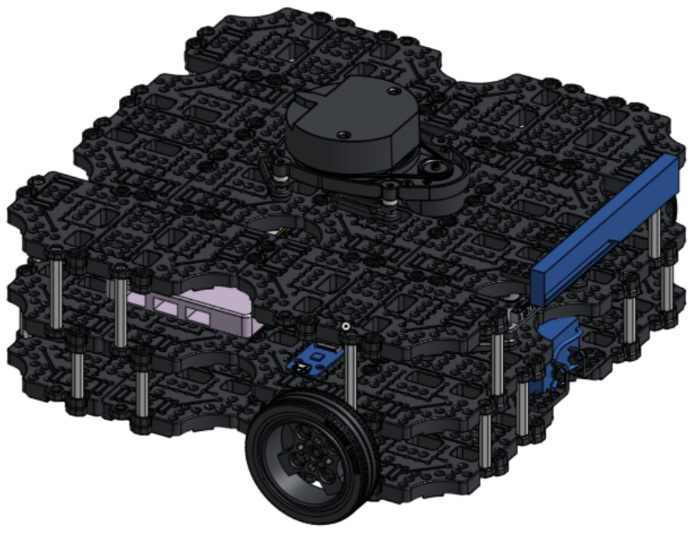


Figure 2.4: CAD of the TurtleBot3 Waffle 3D model.

The dimensions of the TurtleBot3 Waffle is shown in Figure 2.5 [22]. It is developed for use in education and product prototyping, by being small, affordable, programmable and ROS-based. TurtleBot3 Waffle contains a Laser Distance Sensor and an Intel Realsense for 3D perception and is able to run SLAM algorithms to build a map and move around within the environment by the use of these sensors. It can thus be used for mapping and navigation within the environment it's contained in. For the purpose of this project, the TurtleBot3 Waffle is used for SLAM operations, as well as navigating to the points found by the developed inspection algorithm. It is used to evaluate the positions generated from the algorithm in a simulated environment.

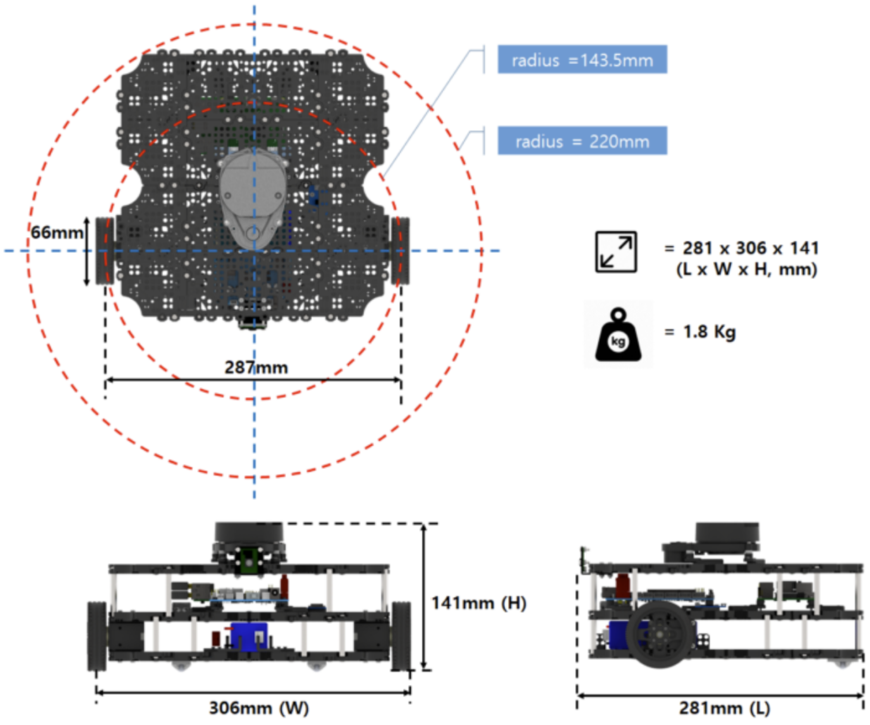


Figure 2.5: Illustration of the dimensions of the TurtleBot3 Waffle model [22].

2.7 Development environment

This section is based on the specialization project [1, p. 9].

2.7.1 Docker

Docker is a container-based framework for developing, deploying and running applications [23]. Docker enables creation of isolated containers which can be run simultaneously on one host, as well as easy sharing of code without infrastructure restrictions. The containers contain everything required to run the application, meaning that developers is not required to have specific software installed on their local device to be able to run the application properly. Each of the modules in the project is implemented as microservices in Docker, enabling easy communication between multiple services in one host.

2.7.2 ROS

Robotic Operating System (ROS) is a software development kit that offers a set of fundamental components for constructing robotic applications [24]. It is designed to be compatible with a wide range of robotic platforms. The primary role of ROS is to facilitate communication between programs within a distributed system, making it particularly useful for creating modularized subprograms (referred to as nodes) that handle specific functionalities of a given problem [25]. In order to achieve effective communication, ROS provides three key communication mechanisms: topics, services, and actions. Topics are utilized for transmitting data streams between nodes, while services and actions establish a client/server architecture. These communication tools are employed by defining message structures that encapsulate the content to be exchanged within the ROS framework.

The project uses ROS1, specifically the ROS noetic distribution, which is mainly targeted to Ubuntu 20.04. In this project, all modules communicate through ROS.

2.7.3 Gazebo

Gazebo is an open-source 3D robot simulation software that facilitates the testing and simulation of robotic systems within a virtual environment [26]. Key features of Gazebo include its capability to execute physics simulations, enabling robots to interact with their surroundings. Furthermore, it offers sensor simulations, which generate sensor data. Additionally, Gazebo provides 3D visualizations that allow for the monitoring of simulated robots and objects. These features collectively make Gazebo a valuable tool for assessing and refining robotic systems prior to real-world deployment.

Gazebo is used in the project to simulate and visualize the 3D environment of the Huldra platform in a virtual environment. Illustrations of the Gazebo visualizations of the Huldra platform can be seen in Appendix C.

2.7.4 Rviz

Rviz (ROS visualization) is a 3D visualization tool, allowing visualization of the state of the robot and sensor data [27]. In addition, it allows placing virtual markers in the virtual environment, which may enable simpler visualization during testing.

Rviz is used in the project to visualize the computed walkway path and the possible inspection positions of the robot, as well as the raw image seen from the camera of the robot. This makes is possible to see partial results, and to follow the robot while it is performing inspections.

3 Method

This chapter describes the accomplishment of the sub-objectives outlined in Section 1.2. It is structured into six distinct sections, where the first three sections present the premises of the algorithm, while the three last sections each address the solutions to the issues raised in the introductory chapter. The first section will describe how the required data is acquired. The second section outlines the assumptions and simplifications the developed algorithm is based upon. The third section illustrates an overview of the modules of the algorithm and their interactions with other modules. The fourth section will describe the integration between the developed algorithm and the testing simulation framework designed by Equinor. The fifth section will discuss the creation of a search algorithm, which aims at reducing the run time compared to the existing algorithm. Finally, the sixth section will incorporate the characteristics of the robot into the algorithm, enabling it to identify optimal inspection positions that do not strictly lie on the walkway line.

3.1 Data acquisition

This section is taken from the specialization project [1, p. 11].

The developed software requires information describing the 3D models where inspection is to be performed [6, p. 10]. This information is acquired from Equinor's digital twins of their industrial facilities, specifically the Huldra platform. As this information is only available through Equinor APIs, which requires access, Equinor has provided the files directly. The files have then been manually entered into the source of the project.

The provided object files contain different information. Some object files represent the 3D model of the platform as a whole, while others only contain information about the 3D structure of the walkway. Furthermore, to enable less resource-demanding testing, these object files have been divided into parts of the platform of different sizes. This allows for testing only small areas while developing the code, while having the opportunity to test larger areas as the algorithm is tested thoroughly.

In addition to object files, the digital twins also have plain text documents that describe the different valves on the platform. The description includes the name of the valve, the coordinates (x, y, z), and the orientation $(roll, pitch, yaw)$ of each valve. Such a description is called a *tag*, and an example of a tag of one valve is illustrated in Listing 3.1.

Listing 3.1: An example of a tag of one valve in the Huldra model.

```
20-20000 VF
"xCoordinate": 304500,
"yCoordinate": 117014,
"zCoordinate": 30016,
roll: 0
pitch: 0
yaw: -pi/2
```

The developed algorithm requires information about the optimal inspection direction of the valve, which is not directly provided by Equinor's APIs [6, p. 12]. The optimal inspection direction is defined as the direction that the camera must be directed in order to perform an inspection. The project acquires this data manually, by observing the direction of each valve in the digital twin and manually entering this into the source code as the optimal inspection direction.

The implementation of the algorithm requires two 3D models: one for the walkway, and one for the entire area of interest. The 3D model of the walkway is converted into 3D coordinates by leveraging the properties of the vertices and faces contained in the 3D representation. Each line in the *obj* file is read by the software and stored for use in the calculation of vertices and faces. The 3D model of the entire area of interest is accessed by using the *Trimesh* library in Python using the `load()` function [13]. This model is then used to calculate the distance, angle and number of obstacles between the camera and the target object, which are used in the optimization process.

For testing purposes, an excerpt of the Huldra platform referred to as *huldra-smaller* is used throughout the project. Figure 6.1 illustrates the excerpt seen from above, while Figure 3.2 illustrates it from the side view. The yellow part represents the walkway line, while the remaining parts are the rest of the facility including the valves to be inspected. The inspector robot is illustrated as a white box.

3.2 Assumptions and simplifications

The algorithm only considers valves and no other sensors. A fundamental premise in the optimization algorithm is that the optimal inspection position is located at the forefront of the valve [1, p. 9]. Specifically, when considering the angle as the only variable in the optimization process, the optimal inspection point is characterized by an angle of zero degrees between the camera direction and the optimal inspection direction for the valve. As the angle deviates from zero, the absolute value of the angle increases. An angle of 90 degrees corresponds to the camera being positioned perpendicular to the side of the valve, while angles greater than 90 degrees place the camera behind the target valve. Nonetheless, this assumption may not hold in reality as different types of valves may necessitate inspections from other viewpoints to provide the required information concerning the valve's condition. An illustration of what a suitable angle and an unsuitable angle for inspection are can be seen in Figure 3.3.

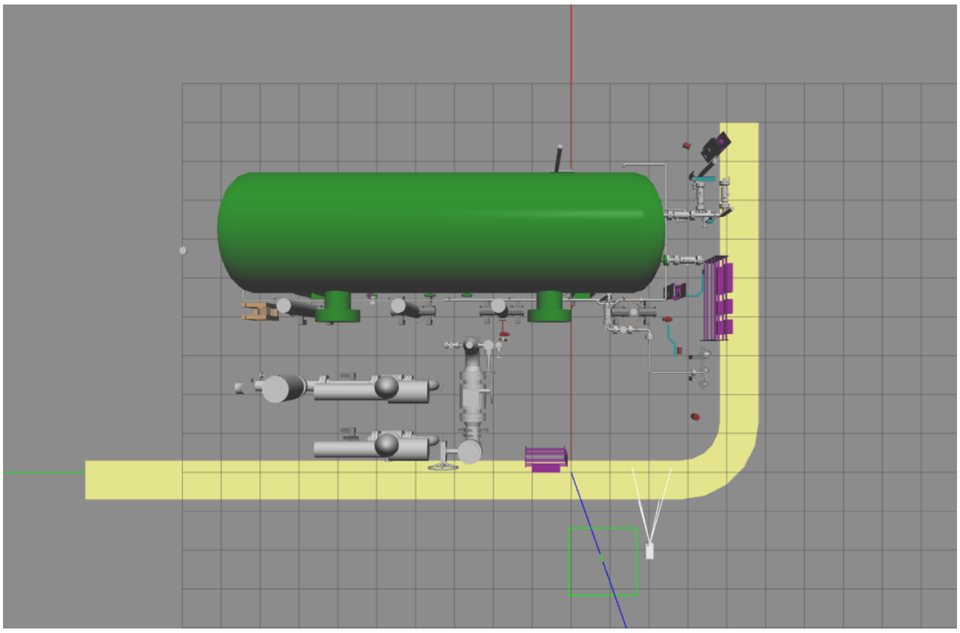
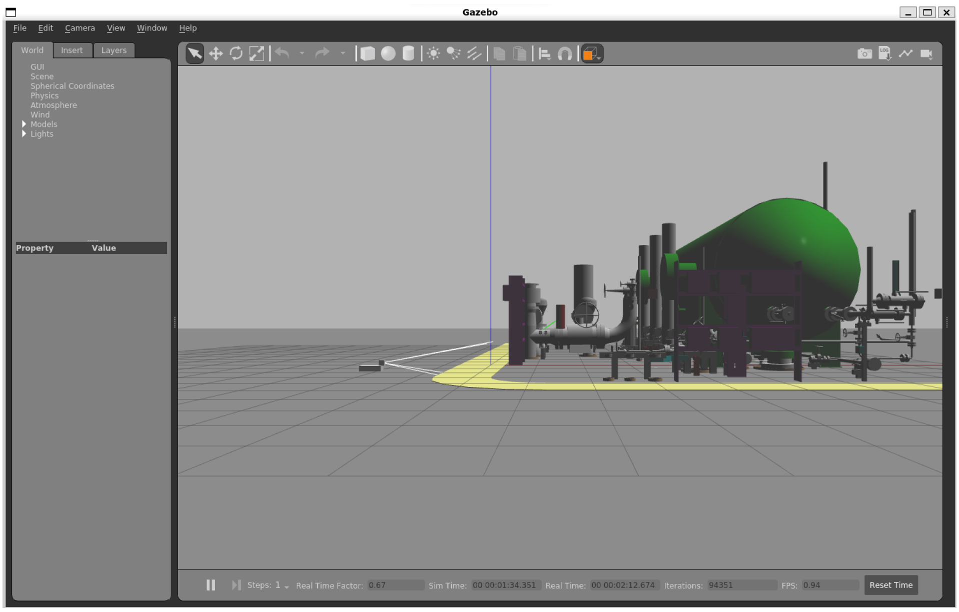


Figure 3.1: Illustration of the digital twin of the excerpt of the Huldra facility referred to as *huldra-smaller* seen from above.

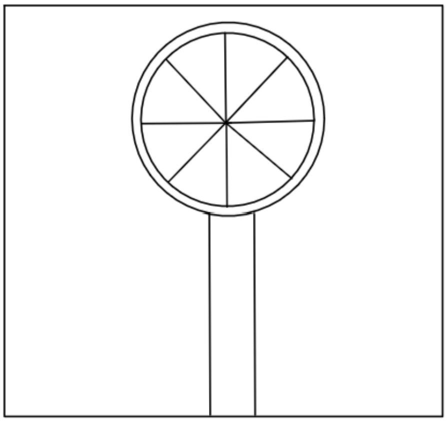


Steps: 1 Real Time Factor: 0.67 Sim Time: 00 00:01:34.351 Real Time: 00 00:02:12.674 Iterations: 94351 FPS: 0.94

Figure 3.2: Illustration of the digital twin of the excerpt of the Huldra facility referred to as *huldra-smaller* seen from the side.

Valve angle illustration

(a) Illustration of a camera directed at the valve with an angle between the camera direction and the optimal inspection direction of zero degrees.



(b) Illustration of a camera directed at the valve with an angle between the camera direction and the optimal inspection direction with an absolute value larger than zero degrees.

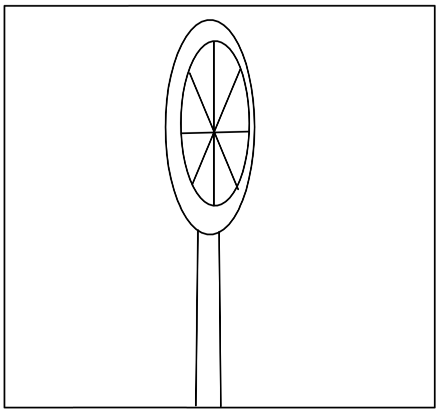


Figure 3.3: Illustration of camera directions toward a valve.

As the Huldra industrial facility is one of few facilities owned by Equinor with open-source digital twins, all testing is restricted to the Huldra platform. Moreover, all experimentation and testing conducted in this study is performed solely in simulated environments. This is primarily due to the unavailability of the Huldra facility, which no longer exists, as well as the lack of available robots for testing in another operational facility.

This means that all testing is performed in a static environment, where all obstacles are predictable. The presence of humans or scaffolding may therefore impact the solutions in reality, which is not considered by the developed algorithm.

3.3 Overview of the algorithm modules

The algorithm consists of five distinct modules, namely ROS, Gazebo, Rviz, robot inspector, and mission planner module. ROS and Gazebo are utilized as the framework for the robot, while Rviz is employed to visualize the results. The mission planner constitutes the centerpiece of the project, as it calculates the optimal inspection positions and is modified and evaluated throughout the project. The robot inspector module is responsible for maneuvering the robot to the predetermined positions given by the optimal inspection positions computed in the mission planner module. The communication flow between these modules is illustrated in Figure 3.4, whereby all communication is mediated by the ROS module.

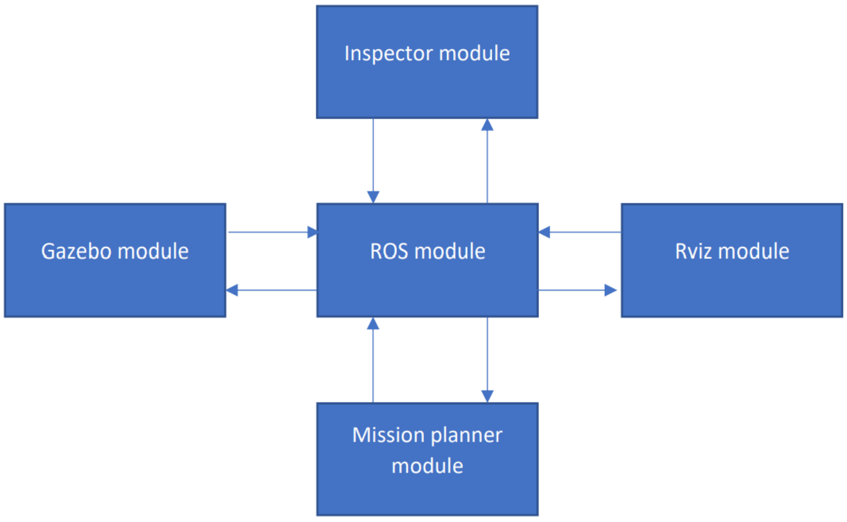


Figure 3.4: Overview of the communication flow between modules [1].

3.4 Integration of planned optimal inspection positions with Equinor's system

Due to the limited availability of robots for testing purposes in physical industrial facilities, the evaluation of the algorithm's performance relies heavily on simulations. However, to enable future testing with physical robots at Equinor's facilities, the algorithm is integrated with Equinor's simulation tools. This integration is facilitated by ISAR (Integration and

Supervisory Control of Autonomous Robots) [28], a framework that enables the issuance of commands to robots for executing missions and obtaining results both within a simulated and physical environment. The utilization of ISAR serves as a pre-deployment testing mechanism for Equinor's robots, offering an opportunity to validate the algorithm's output before actual deployment in physical facilities, and has demonstrated a high degree of accuracy when compared to subsequent testing conducted at the physical facility. As this is how the robot missions are tested before deployment at Equinor's facilities, it is valuable to incorporate the algorithm's generated optimal positions into the simulation tool, thereby facilitating comprehensive evaluation and refinement of the algorithm's performance.

3.4.1 Implementation

To run the full ISAR system, a compatible robot that meets the interface requirements of the ISAR framework must be installed. Equinor has implemented an ISAR version of the TurtleBot3 Waffle, which enables running the TurtleBot with ISAR through simulations [29]. As the ISAR framework is also used to control the physical robots in the actual industrial facilities, the gap between testing on simulations and on a physical robot is minimal, making the integration with ISAR essential in testing the algorithm in reality.

The integration of the developed algorithm into Equinor's system requires following guidelines provided in [28] and [29]. On the TurtleBot side of the integration, the selected Gazebo model (i.e. the digital twin) and a corresponding model map must be added to the isar-turtlebot code base. This involves placing the Huldra facility's obj-file into the "models/huldra-smaller" folder and adding a "huldra-smaller.world" file to the "worlds/" folder. The world file is represented by an *.sdf* file, as illustrated in Listing 3.2. A *.sdf* file is a specified description of a simulation that is not closely coupled with specific simulators, and is therefore versatile in use [30]. There are several *tags* used to describe the simulation, as indicated in Listing 3.2 within the "<>" operators. The $< pose >$ tag moves the model from the digital twin to the origin by specifying the position and orientation the model must be moved in order to reach the origin. The $< visual >$ tag describes the visual properties of the link, such as shape and size for visualization purposes, whereas the $< collision >$ tag describes the collision properties of the link. The $< geometry >$ tag describes the shape of the visual or collision object. In this project, it is represented by the object file of the Huldra model.

Listing 3.2: World file of the Huldra facility..

```
<?xml version='1.0'?>
<sdf version='1.4'>
    <model name='huldra-smaller'>
    <pose>119 307 -29.450 1.571 0 0</pose>
        <static>true</static>
        <link name='link0'>
            <visual name="visual0">
                <pose>0 0 0 0 0 0</pose>
                <geometry>
                    <mesh>
                        <uri>file://huldra-smaller/meshes/
                        huldra-smaller.obj</uri>
                    </mesh>
                </geometry>
                <cast_shadows>0</cast_shadows>
            </visual>
            <collision name="collision0">
            <geometry>
                <mesh>
                <uri>file://huldra-smaller/meshes/
                huldra-smaller.obj</uri>
                </mesh>
            </geometry>
            </collision>
        </link>
      </model>
</sdf>
```

Additionally, a default configuration for the robot's initial pose in the simulation must be defined in the "config/huldra-smaller.cfg" file. To create a map of the Huldra environment based on the world file, the tutorial in [31] is followed. The map is created by first running the isar-turtlebot repository. To launch the simulation world, which is the digital twin of the Huldra facility, several commands need to be executed as delineated in Listing 3.3.

Listing 3.3: Launching the simulation world from terminal.

```
#!/bin/bash
source /opt/ros/noetic/setup.bash
source /home/catkin_ws/devel/setup.bash
export TURTLEBOT3_MODEL=waffle
roslaunch isar_turtlebot gazebo_turtlebot.launch
world_name:=huldra-smaller
```

Subsequently, the process of starting the SLAM node involves accessing the running container by executing the command given in Listing 3.4.

Listing 3.4: Entering the running container from terminal.

```
docker exec -it turtle_sim bash
```

and the node is started by running the command given in Listing 3.5.

Listing 3.5: Running the SLAM node from terminal.

```
export TURTLEBOT3_MODEL=waffle
roslaunch turtlebot3_slam turtlebot3_slam.launch
slam_methods:=gmapping
```

The subsequent stage involves initiating the teleoperation node, which is enabled by accessing the container in a new terminal and executing the commands outlined in Listing 3.6.

Listing 3.6: Running the Teleoperation node from terminal.

```
export TURTLEBOT3_MODEL=waffle
roslaunch turtlebot3_teleop
turtlebot3_teleop_key.launch
```

This enables one to control the TurtleBot within the environment and facilitate its mapping of the surroundings. Upon being content with the map, it can be saved by launching another terminal and executing the command given in Listing 3.7.

Listing 3.7: Saving the generated map from terminal.

```
rosrun map_server map_saver -f ~/map
```

The execution of this command generates two file formats of the map, namely a *.pgm* file and a *.yaml* file, that represent the environment mapped by the TurtleBot.

Finally, the inspection positions generated must be converted into a *json* file of the format required by the ISAR framework. The required format of the mission definition is constructed by a dictionary that contains information about the task to be performed. The required format is illustrated by an example in Listing 3.8. The dictionary contains the id of the mission, as well as the task that is to be performed. The task includes the steps required to get the image taken, namely driving to the pose described by the optimal inspection positions (position and orientation of the robot) and an instruction to take the image of the target valve by providing the position coordinates of the target valve. Each position and orientation is described within a specified frame, namely the asset frame.

Listing 3.8: Example of a mission dictionary used by the ISAR framework.

```
example_mission_dict = {
"id": id,
"tasks": [
{
    "steps": [
        {
            "type": "drive_to_pose",
            "pose": {
                "position": {
                    "x": 2,
                    "y": 3
                    "z": 1,
                    "frame": {"name": "asset"},
                },
                "orientation": {
                    "x": 0,
                    "y": 0,
                    "z": 0.782,
                    "w": 0.782,
                    "frame": {"name": "asset"},
                },
                "frame": {"name": "asset"},
            },
        },
        {
            "type": "take_image",
            "target": {"x": 3, "y": 0, "z": 1,
            "frame": {"name": "asset"}},
        },
    ],
    },
],
}
```

The implementation of providing the optimal inspection positions calculated by the developed algorithm to the required format is given in Listing 3.9.

Listing 3.9: The implementation of the required format of the dictionary in the algorithm.

```
    mission_dict = {
    "id": id,
    "tasks": [
    {
        "steps": [
            {
                "type": "drive_to_pose",
                "pose": {
                    "position": {
                        "x": inspection_pose.position.x,
                        "y": inspection_pose.position.x,
                        "z": inspection_pose.position.z,
                        "frame": {"name": "asset"},
                    },
                    "orientation": {
                        "x": inspection_pose.orientation.x,
                        "y": inspection_pose.orientation.y,
                        "z": inspection_pose.orientation.z,
                        "w": inspection_pose.orientation.w,
                        "frame": {"name": "asset"},
                    },
                    "frame": {"name": "asset"},
                },
            },
            {
                "type": "take_image",
                "target": {"x": poi_point.orientation.x,
                "y": poi_point.orientation.y,
                "z": poi_point.orientation.z,
                "frame": {"name": "asset"}},
                },
            ],
            },
        ],
    }
```

Then, the dictionary of the mission tasks is stored within a *json* file to be provided to the ISAR framework. The implementation is done by exploiting Python's *json* library and is illustrated in Listing 3.10. For each inspection point, a *json* file is created with the id, which corresponds to a specific valve, of the mission task in the *json* file name. This way, one can inspect the valves of interest and not be limited to testing each valve in the facility each time.

Listing 3.10: Implementation of storing the dictionary to a json file.

```
outfile = open("./inspection_pose"+str(id)+".json",
          "w", encoding ='utf8')
json.dump(mission_dict, outfile)
outfile.close()
```

Lastly, the *json* files containing the missions are manually placed inside the folder **isar/src/config/predefined_missions** of the ISAR framework [28].

3.4.2 Testing Equinor's simulation tool

Testing Equinor's simulation tool is done by connecting the isar and isar-turtlebot repositories and providing the missions id to the isar framework. This enables the TurtleBot to move to the given location provided by the optimal inspection points, causing the possibility of detecting whether the algorithm provides suitable inspection points in reality, assuming that the digital twin is an accurate representation of the physical facility.

Firstly, the isar-turtlebot repository is built by issuing the command ”docker compose build” from the terminal, and giving the docker container access to the screen by running ”xhost +local:‘docker inspect –format=' .Config.Hostname ' turtle_sim‘” [29]. The desired simulation world is set by changing the `WORLD_NAME` variable defined in the entrypoint file. To run the huldra-smaller facility on isar-turtlebot, one may then issue the command ”WORLD_NAME=huldra-smaller docker compose -f docker-compose.yml -f docker-compose-nvidia.yml up - -build”.

While the isar-turtlebot is running, ISAR using the turtlebot as the robot is started by ”docker-compose -f docker-compose-turtlebot.yml up - -build” [28].

3.5 Developing the search algorithm

As mentioned in Section 1.1, the algorithm previously developed suffers from fundamental deficiencies regarding its run time and scalability. Mitigating the analysis of extraneous points that do not provide necessary information about the optimal inspection point, such as points located far from the target valve or from an unfavorable angle, will improve the algorithm's run time and consequently its scalability. Therefore, this thesis suggests implementing a search algorithm to improve the algorithm's efficacy.

3.5.1 Implementation of search algorithm

The first to consider is where to start the search to find the optimal inspection position for a target valve. As identified in the specialization project [1], three main factors determine whether an inspection point is considered optimal: the distance between the inspection point and the target valve, the angle between the direction of the camera and the optimal inspection direction of the target valve, and whether there is a clear view from the inspection point towards the target valve. In finding a suitable starting position for the search algorithm, the distance measure is utilized by locating the point on the walkway closest to the target valve. Finding the distance between the inspection point and the valve is done by calculating the Euclidean distance between the possible inspection point and the target point of the valve, i.e., computing

$$d(\mathbf{p},\mathbf{q}) = \sqrt{\sum_{i}^{n}(q_i - p_i)^2}$$

where **p** and **q** are the n-space coordinates of the points to be calculated the distance between [1, p. 14]. The implementation is given in Listing 3.11. The inspection point on the walkway with the shortest distance to the target valve is then chosen as the starting search point of the search algorithm, and the calculated distance is temporarily set as the score of the current search point.

Listing 3.11: Implementation of finding the distance between the inspection point and the target point of the valve.

```
def get_distance_between_points(p1: Point, p2: Point):
    distance = sqrt((p2.x-p1.x)**2
                + (p2.y-p1.y)**2
                + (p2.z-p1.z)**2)
    return distance
```

After the starting point of the search has been found, the angle between the camera's direction and the target valve's optimal inspection direction is considered. Considering that the optimal angle when disregarding other factors equals an angle of zero degrees (i.e. the camera faces the target valve directly from the front of the valve), the larger the absolute value of the angle is, the less probable it will be to get an image that is informative enough to assess the state of the valve. Valves that have an angle between their optimal inspection direction and the direction of the camera above 90 degrees imply that the camera is located behind the valve and one cannot see the front of the valve. As the angle decreases, the camera is placed closer to the optimal inspection direction yielding the most informative images. This concept, that a smaller angle yields better positions than larger angles, is the second factor the search algorithm exploits to determine the optimal inspection position.

Starting with the search point found by the minimum distance principle, the algorithm calculates the angle between the search position direction and the optimal inspection direction of the target valve, and adds the absolute value of the angle times 100 to the temporary score of the inspection point. Based on the optimal inspection direction of the valve, the algorithm calculates the movement of the search point along the orientation of the target valve, moving the search point in the direction of the front of the target valve. This is done by exploiting the quaternion that describes the orientation of the target valve. As described in Section 2.4, this is done by pre multiplying the current inspection point by the quaternion, and post multiplying the current inspection point by the conjugate of the quaternion. However, this may cause the new inspection point to be dislocated from the walkway line. As the desired inspection points should be located on the walkway by restrictions on the robot's movement, the algorithm proceeds by projecting the new search point onto the walkway line by calculating the point on the walkway with the smallest distance to the calculated search point. The implementation is shown in Listing 3.12. It exploits the Python built-in functions of quaternions, namely the *tf.transformations* library and specifically the functions `quaternion_multiply(quarternion1,quarternion0)`, which returns the multiplication of two quaternions, and `quarternion_conjugate(quarternion)`, which returns the conjugate of a quarternion [32]. The search point is required to be converted to the form of a quaternion to enable the use of these functions and is converted back to a `Point()` object after the quaternion calculations are performed.

Listing 3.12: Implementation of finding the next search point.

```
 #Update temporary search variable
    temp_search_point = [current_search_point.x,
                          current_search_point.y,
                          current_search_point.z,
                          0]
    temp_search_point = quaternion_multiply(
                         poi_orientation ,
                         temp_search_point)
    temp_search_point = quaternion_multiply(
                         temp_search_point ,
                         quaternion_conjugate(
                         poi_orientation))
    temp_search_point = Point(temp_search_point[0],
                         temp_search_point[1],
                         temp_search_point[2])

#Project temporary search variable onto walkway
    for i in range (0,len(possible_inspection_points)):
        distances[i] = get_distance_between_points(
                         possible_inspection_points[i],
                         temp_search_point)
        if distances[i] < min_distance:
            min_distance = distances[i]
            current_search_point = possible_inspection_points[i]
```

For the new search point, the angle between the camera direction and the optimal inspection direction is calculated and added to the scoring function multiplied by 100, indicating that the angle is weighted relatively higher than the distance. The coordinates and corresponding scores are stored within a *pandas* dataframe on the form given in Table 3.1. Pandas is a Python library for performing data analysis [33], and is in the project used as a tool for generating a dataframe, enabling simple manipulation of the data in the dataframe.

Table 3.1: The structure of the pandas dataframe used to store the potential inspection points.

Index	Coordinate	Score
0	(x_1, y_1, z_1)	number
1	(x_2, y_2, z_2)	number
...	...	...

This is done within a while loop that terminates when the absolute value of the angle of the last search point is greater than the absolute value of the angle of the current search point, creating a dataframe of possible inspection points and discarding uninteresting points. The third factor for inspecting a valve, namely, if there is a clear view between the camera and the target valve, is tested on each point in the dataframe of the potential inspection points. The score is updated by adding the number of obstacles between the camera and the target valve times 100 to weigh it relatively higher than the distance between the camera and the target. The implementation is given in Listing 3.13.

Listing 3.13: Implementation of finding the obstacles between the camera and target valve.

```
for i in range(0,len(searches_df.index)):
    current_search_point = searches_df['Coordinate'][i]
    current_search_point_score = searches_df['Score'][i]
    obstacles_between_count = self.get_obstacles_between(
                                current_search_point,
                                poi_point, self.mesh_file)
    current_search_point_score += 100*obstacles_between_count
    searches_df.at[i, 'Score'] = current_search_point_score
```

Finally, the dataframe is sorted based on the scores, from low to high. The optimal inspection point is thus found by extracting the coordinates of the point with the lowest score, that is, from the top row of the sorted dataframe. This results in the minimization of the combined objective function when desiring a low distance between the camera and target valve, a small angle between the camera direction and the optimal inspection direction, and a low number of obstacles between the camera and target valve.

The search algorithm is thus a local optimization technique for finding the optimal inspection position. In short, the algorithm decides on the starting point by utilizing the minimum distance between the target valve and walkway points, proceeding with the search based on the angle between the camera direction and the optimal inspection direction of the target valve and the number of obstacles between the camera and target valve.

3.5.2 Testing the search algorithm

Given that the implementation of the search algorithm aims to reduce the run time, the tests were conducted by running the developed search algorithm with different resolutions on the *huldra-smaller* model. The term *resolution* will throughout the rest of the report refer to the number of samples of points produced along the walkway line, that is, the number of possible inspection points. The resolutions of $5m, 1m, 0.5m$ and $0.1m$ were tested, and the run time and the number of positions that yielded suitable inspections were recorded.

3.6 Integration of robot characteristics

As not all valves are visible from the walkway, not all valves can be inspected with the camera located on the walkway. Moving the camera beyond the walkway line by exploiting the robot characteristics and thereby creating new possible camera inspection positions while the robot is still contained within the walkway may lead to positions where it is possible to inspect the valves that are not visible from the walkway.

3.6.1 Introduction of instructions of placement of camera with respect to robot characteristics

Formulating a set of instructions outlining the process of camera placement can be approached from two perspectives. The first perspective involves considering the robot as a given entity, along with its inherent characteristics, and subsequently determining an optimal camera position based on its range of motion. Conversely, the second perspective involves computing the optimal positions for valve inspection and subsequently selecting a robot with characteristics that enable it to reach these locations. The Equinor team has advised exploring the latter alternative, which will therefore be the focus in this thesis.

In order to reach an algorithm that incorporates the robot characteristics one must identify the characteristics of the robot that impacts the camera placement. This includes the size, shape and range of motion of the robot. As the purpose is to be able to move the camera with respect to the robot's placement within the walkway, the focus is limited to the range of motion of the robot. The work will consider a robot that is able to move the camera in the positive vertical direction. The robot can thereby be described by the robot body, which is contained within the walkway, and a camera mounted on an arm that can only be moved in the vertical direction. An illustration of such a simple robot can be seen in Figure 3.5.

Introducing a camera mounted on a robot arm in the vertical direction instead of viewing the camera and robot as a single entity implies that the robot have one extra degree of freedom compared to before. The robot is now able to move within the (x, y, z)-space instead of being restricted to a given z-parameter by the walkway and moving within the (x, y)-plane.

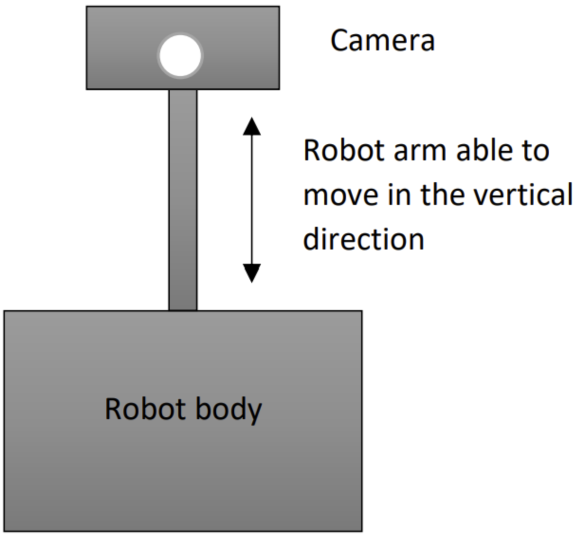


Figure 3.5: Illustration of a robot with movable vertical positioning of the camera.

3.6.2 Implementation of robot characteristics into the algorithm

The search algorithm provides several points that lie on the walkway line within a certain range of the valve that may be used for inspection, where each possible inspection point is at a constant height of $0.1m$ above the ground floor. For each of these points, there may be points in the vertical direction that provide better viewpoints than those lying on the walkway. The approach used in the project is to exploit the points already found by the search algorithm, and sampling points in the vertical direction based on a given interval of points. Specifically, the search algorithm stores multiple possible inspection points and scores each point based on the aforementioned measures in a dataframe. The dataframe of the possible points is then iterated through, and for each of the possible inspection points, sampled points in the vertical direction provided by the height interval are added to the dataframe and subject to collision detection. The given interval of points to be sampled describes the robot's characteristics, and there must be decided upon a robot that is able to reach the points given by the interval.

Listing 3.14 shows the implementation of the robot characteristics in a simplified manner. The presence of the "..." represents the existence of additional code performing other functions, as the robot characteristics implementation are intertwined into the search algorithm described in Section 3.5. It starts by defining the interval of the samples in the vertical direction by using the *NumPy* library's `np.linspace` function, which returns a list of evenly spaced numbers over a given interval [34]. After entering the *while*-loop, it exploits the current search point found by the search algorithm added to the *pandas* dataframe of possible inspection points and computes the z-coordinates based on the given interval. The result is then appended to the dataframe of possible inspection points.

Listing 3.14: Implementation of robot characteristics.

```
height interval = np.linspace(0,1,5)
...
while abs(current_ange) < abs(last_angle):
...
    for i in range(1,len(height_interval)):
    current_search_point_vertical = Point(
                                    current_search_point.x,
                                    current_search_point.y,
                                    current_search_point.z+
                                    height_interval[i]
                                    )
                 new_row = pd.DataFrame([
                 [current_search_point_vertical ,
                 current_search_point_score]],
                 columns=['Coordinate', 'Score'])
                 searches_df = pd.concat([searches_df ,
                 new_row]).reset_index(drop=True)
```

Subsequently, the dataframe of possible inspection points, which now also includes points above the walkway plane, is inspected for obstacles between the camera and target valve as described in Section 3.5.1.

3.6.3 Testing the algorithm including the robot characteristics

Four tests were conducted to evaluate the search algorithm's performance after incorporating the robot characteristics. These tests involved varying resolutions in the horizontal plane and vertical height intervals.

In the first, second, and third tests, a resolution of $0.1m$ was employed, while the height intervals were adjusted. The first test utilized a height interval in the z-direction ranging from $0m$ to $1m$, with an evenly distributed set of five samples within this interval. The second test expanded the height interval to span from $0m$ to $2m$, maintaining the same distribution of five samples across the interval. On the other hand, the third test maintained a height interval of $0m$ to $1m$, but increased the number of evenly distributed samples to ten.

The fourth test utilized a resolution of $5m$ and a height interval from $0m$ to $1m$. In this case, five evenly distributed samples were obtained along the vertical direction.

An overview of the conducted tests can be seen in Table 3.2.

Table 3.2: Overview of results when incorporating robot characteristics into the search algorithm.

Test number	Resolution	Height interval	Number of samples within height interval
1	$0.1m$	$0m$ to $1m$	5
2	$0.1m$	$0m$ to $2m$	5
3	$0.1m$	$0m$ to $1m$	10
4	$5m$	$0m$ to $1m$	5

4

Result

The following chapter is divided into three sections, where each subobjective is considered individually in each section. An overview of the corresponding valve numbers and names used throughout the rest of the report is given in Appendix A, and an overview of the valve visibility from the walkway is given in Appendix B.

4.1 Simulation in Equinor's simulator system

4.1.1 Map generation

The map generated as described in Section 3.4.1 is depicted in Figure 4.1. It provides essential information regarding the robot's trajectory and visual observations, enabling ISAR to effectively guide and maneuver the robot to the designated inspection positions. This information serves as a crucial input for ISAR's decision-making processes, ensuring that the robot reaches the correct locations as determined by the generated inspection positions. The dark gray area represents unknown environment, while the light grey area represents collision free areas and the black areas represent occupied areas.

4.1.2 Simulations

Regrettably, the planned test could not be carried out as scheduled due to unforeseen complications arising from the ISAR code provided by Equinor. Despite the preparations, it was discovered that the code obtained from Equinor contained critical errors or inconsistencies that rendered it unsuitable for conducting the intended test.

The issues encountered with the code's functionality significantly hindered the ability to execute the test as the provided code contained errors preventing it to build and be executed. Efforts were made to collaborate with the external company to address and resolve the code-related concerns, but unfortunately, the necessary adjustments could not be implemented within the timeframe of this study.

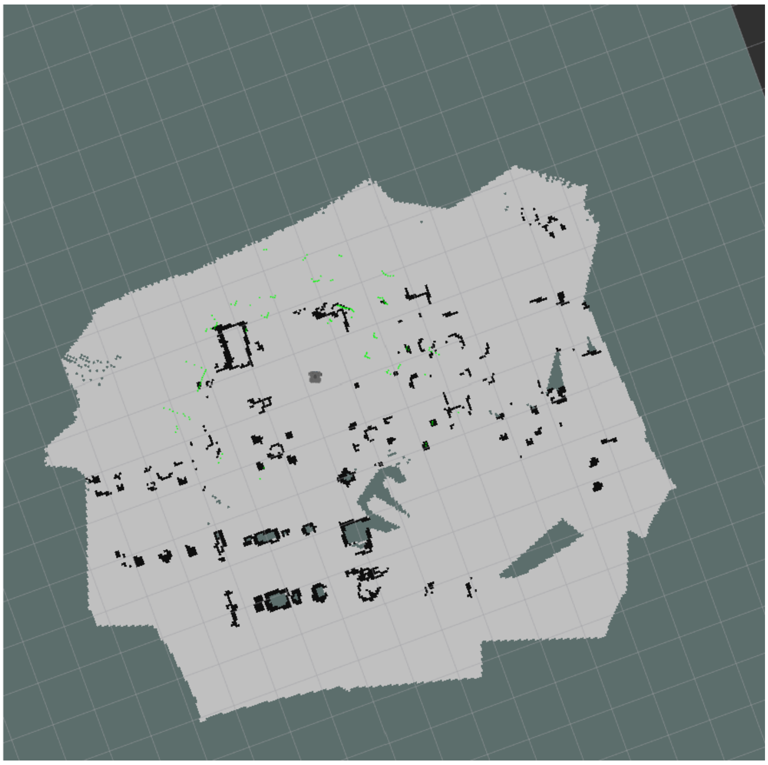


Figure 4.1: The map generated of the *huldra-smaller* facility. The dark gray area represents unknown environment, while the light grey area represents collision free areas and the black areas represent occupied areas.

The inability to conduct the test has prevented the assessment of the search algorithm's real-life performance. Had the test been carried out, it would have revealed whether the generated optimal inspection points were suitable in simulations and if they would translate into suitable positions within the physical facility, assuming the digital twin accurately represents the facility. Nevertheless, since the digital twin utilized for identifying the optimal inspection positions is also used by the simulation environment, it is reasonable to anticipate that it will yield comparably satisfactory results as the executed Gazebo simulations.

4.2 Result of search algorithm

As stated earlier in the report, the main purpose of implementing a search algorithm aimed at finding optimal inspection positions is to improve the efficiency and scalability of the algorithm, i.e. reducing the run time when finding the optimal inspection points. The performed tests evaluate the properties of the run time and accuracy obtained by the algorithm. The accuracy is a measure of the number of valves the algorithm found suitable inspection positions for. Inspection positions yielding images where the entire front of the valve can be seen are labeled suitable for inspection, while positions yielding images where one may inspect partly the front of the valve are labeled adequately suitable for inspection. Otherwise, the inspection positions were considered unsuitable for inspection purposes. Each image produced by the algorithm was inspected manually to determine whether the positions produced suitable or unsuitable images for inspections.

As outlined in Section 3.5.2, a series of four tests were conducted to evaluate the search algorithm. The corresponding test results are presented in Table 4.1, Table 4.2, Table 4.3 and Table 4.4 for the resolutions of $5m$, $1m$, $0.5m$ and $0.1m$, respectively. It was observed that for valve numbers $5, 6, 8, 9$ and 10, the algorithm failed to identify a suitable inspection position across all tested resolutions. However, the algorithm consistently identified appropriate inspection positions for valve numbers $1, 12, 13$ and 14, irrespective of the resolution used. For valve numbers 0, 1, 2 and 14, the respective images taken from the optimal inspection points for the different resolutions are given in Figure 4.2, Figure 4.3, Figure 4.4 and Figure 4.6, respectively.

Figure 4.2 shows the images taken by the inspector robot of valve number 0, where each subfigure is the result of a given resolution. It is observed that all images taken of the target valve number 0 do not provide a suitable inspection position for taking an image of the valve, as one is not able to see the target valve in the images. However, one may note that each image is taken from different positions based on the resolution provided to the search algorithm.

The images captured by the inspector robot for valve number 1 are depicted in Figure 4.3. Notably, each image exhibits the target valve positioned at the center of the image, with its orientation aligned directly toward the camera. This signifies that the algorithm successfully determined the optimal inspection position for valve number 1, regardless of the resolutions.

In the case of valve numbers $2, 4, 7$ and 11, variations were observed among the test outcomes. Figure 4.4 show the generated images for valve number 2 when employing the aforementioned resolutions. It is evident from Figure 4.4a and Figure 4.4d that both images

Table 4.1: Result from the test performed on the *huldra-smaller* excerpt of the Huldra facility when using resolution $5m$. The resulting images for valve number 0, 1, 2 and 14 is illustrated in Figure 4.2a, Figure 4.3a, Figure 4.4a and Figure 4.6a, respectively.

Valve number	Result of resolution $5m$	Comment
0	Unsuitable	Obstacle preventing the view of the valve
1	Suitable	Clear view of the entire valve from the front
2	Suitable	Clear view of the entire valve from the front
3	Unsuitable	Image taken of the rear of the valve
4	Unsuitable	Obstacle preventing the view of the valve
5	Unsuitable	Obstacle preventing the view of the valve
6	Unsuitable	Obstacle preventing the view of the valve
7	Unsuitable	Obstacle preventing the view of the valve
8	Unsuitable	Image taken perpendicular to the valve side
9	Unsuitable	Obstacle preventing the view of the valve
10	Unsuitable	Most of the valve is covered by an obstacle
11	Suitable	Clear view of the entire valve from the front
12	Suitable	Clear view of the entire valve from the front
13	Suitable	Clear view of the entire valve from the front
14	Suitable	Clear view of the entire valve from the front

Table 4.2: Result from the test performed on the *huldra-smaller* excerpt of the Huldra facility when using resolution $1m$. The resulting images for valve number 0, 1, 2 and 14 is illustrated in Figure 4.2b, Figure 4.3b, Figure 4.4b and Figure 4.6b, respectively.

Valve number	Result of resolution $1m$	Comment
0	Unsuitable	Obstacle preventing the view of the valve
1	Suitable	Clear view of the entire valve from the front
2	Unsuitable	Obstacle preventing the view of the valve
3	Unsuitable	Image taken perpendicular to the valve side
4	Unsuitable	Image taken of the rear of the valve
5	Unsuitable	Image taken perpendicular to the valve front
6	Unsuitable	Obstacle preventing the view of the valve
7	Unsuitable	Obstacle preventing the view of the valve
8	Unsuitable	Obstacle preventing the view of the valve
9	Unsuitable	Obstacle preventing the view of the valve
10	Unsuitable	Obstacle preventing the view of the valve
11	Unsuitable	Obstacle preventing the view of the valve
12	Suitable	Clear view of the entire valve from the front
13	Suitable	Clear view of the entire valve from the front
14	Suitable	Clear view of the entire valve from the front

exhibit the target valve centered within the image, with its orientation directed towards the camera. This indicates that the algorithm successfully identified suitable inspection positions for valve number 2 using resolutions of $5m$ and $0.1m$. Conversely, Figure 4.4b

Table 4.3: Result from the test performed on the *huldra-smaller* excerpt of the Huldra facility when using resolution $0.5m$. The resulting images for valve number 0, 1, 2 and 14 is illustrated in Figure 4.2c, Figure 4.3c, Figure 4.4c and Figure 4.6c, respectively.

Valve number	Result of resolution $0.5m$	Comment
0	Unsuitable	Obstacle preventing the view of the valve
1	Suitable	Clear view of the entire valve from the front
2	Unsuitable	Obstacle preventing the view of the valve
3	Unsuitable	Image taken of the rear of the valve
4	Suitable	Clear view of the entire valve from the front
5	Unsuitable	Image taken perpendicular to the valve side
6	Unsuitable	Obstacle preventing the view of the valve
7	Unsuitable	Obstacle preventing the view of the valve
8	Unsuitable	Obstacle preventing the view of the valve
9	Unsuitable	Obstacle preventing the view of the valve
10	Unsuitable	Obstacle preventing the view of the valve
11	Unsuitable	Obstacle preventing the view of the valve
12	Suitable	Clear view of the entire valve from the front
13	Suitable	Clear view of the entire valve from the front
14	Suitable	Clear view of the entire valve from the front

Table 4.4: Result from the test performed on the *huldra-smaller* excerpt of the Huldra facility when using resolution $0.1m$. The resulting images for valve number 0, 1, 2 and 14 is illustrated in Figure 4.2d, Figure 4.3d, Figure 4.4d and Figure 4.6d, respectively.

Valve number	Result of resolution $0.1m$	Comment
0	Unsuitable	Most of the valve is covered by an obstacle
1	Suitable	Clear view of the entire valve from the front
2	Suitable	Clear view of the entire valve from the front
3	Unsuitable	Image taken of the rear of the valve
4	Suitable	Clear view of the entire valve from the front
5	Unsuitable	Image taken perpendicular to the valve side
6	Unsuitable	Obstacle preventing the view of the valve
7	Adequately suitable	May inspect parts of the valve
8	Unsuitable	Obstacle preventing the view of the valve
9	Unsuitable	Obstacle preventing the view of the valve
10	Unsuitable	Obstacle preventing the view of the valve
11	Unsuitable	Obstacle preventing the view of the valve
12	Suitable	Clear view of the entire valve from the front
13	Suitable	Clear view of the entire valve from the front
14	Suitable	Clear view of the entire valve from the front

and Figure 4.4c demonstrate that the images produced using resolutions of $1m$ and $0.5m$ did not yield appropriate inspection positions as one is not able to identify the target valve in the image, rendering them unsuitable for inspection purposes.

An overview of the results of the tests described in Section 3.5.2 are given in Table 4.5, indicating the performed test, run time of the simulations and the accuracy of the algorithm. It is noteworthy that the highest level of accuracy was achieved when the resolution was set to $0.1m$, although this configuration also exhibited the slowest run time. Notably, adjusting the resolution parameter does influence the computational time of the algorithm, with higher resolutions leading to faster execution.

Table 4.5: Overview of the results obtained from the search algorithm indicating the run time and accuracy of the search.

Test	Run time	Accuracy
Resolution $5m$	00:05:56	6/15
Resolution $1m$	00:11:15	4/15
Resolution $0.5m$	00:15:11	5/15
Resolution $0.1m$	00:27:49	7/15

4.2.1 Comparison to the brute force approach

From the specialization project [1], the brute force approach by evaluating the distance, angle and clear view on every possible inspection point along the walkway yielded the results given in Table 4.6, with two different resolutions, namely of $0.1m$ and $1m$. As seen from the table, when performing tests with resolutions $1m$ and $0.1m$ both yield suitable inspection positions for eight of fifteen valves, whereas the simulation time varies tremendously. When using resolution $0.1m$, the algorithm uses over eight hours to find inspection positions for all fifteen valves, while when using resolution $1m$, the algorithm uses approximately one hour to produce the optimal inspection points for the same valves. As identified in the specialization project [1, p. 27], the ray tracing scheme to check that the camera has a clear view towards the target valve is the main reason for the difference in simulation time in the two cases. Since the number of sampled points along the walkway is directly related to the resolutions, each clear view test using ray tracing is done 10 times more when using resolution $0.1m$ than $1m$, causing the differences in simulation times.

Table 4.6: The results obtained from the specialization project [1].

Test	Simulation time	Accuracy
Resolution 0.1	08:13:50	8/15
Resolution 1	01:03:57	8/15

Upon comparing Table 4.5 and Table 4.6, it becomes apparent that the brute force approach consistently achieves higher levels of accuracy compared to the results obtained through the search algorithm. Conversely, the simulation times for all tests conducted in the brute force approach are considerably longer in comparison to those achieved using the search algorithm.

4.3 Result of incorporating robot characteristics into the algorithm

This section presents the results of the tests conducted as described in Section 3.6.3. The outcomes of test numbers one, two, three and four are presented in Table 4.7, Table 4.8, Table 4.9 and Table 4.10, respectively. Figure 4.5 shows the images taken of valve number 14 for all performed tests when incorporating the robot characteristics.

It is observed that regardless of the resolution and height interval tested, the algorithm successfully identified suitable inspection positions for the same three valves, namely valve numbers $1, 2$ and 13. Additionally, test number one and three both determined suitable inspection positions for the exact same valves, namely valve numbers $1, 2, 4$, and 13, resulting in the lowest accuracy among the four tests. Test numbers two and four also discovered a suitable inspection position for valve number 12. Notably, test number four attained the highest accuracy by identifying six of the generated positions as suitable for inspection purposes.

Table 4.7: Result from test number one performed on the *huldra-smaller* excerpt of the Huldra facility when incorporating robot characteristics into the search algorithm.

Valve number	Result	Comment
0	Unsuitable	Most of the valve is covered by an obstacle
1	Suitable	Clear view of the entire valve from the front
2	Suitable	Clear view of the entire valve from the front
3	Unsuitable	Image taken of the rear of the valve
4	Suitable	Clear view of the entire valve from the front
5	Unsuitable	Obstacle preventing the view of the valve
6	Unsuitable	Obstacle preventing the view of the valve
7	Unsuitable	Obstacle preventing the view of the valve
8	Unsuitable	Obstacle preventing the view of the valve
9	Unsuitable	Obstacle preventing the view of the valve
10	Unsuitable	Obstacle preventing the view of the valve
11	Unsuitable	Obstacle preventing the view of the valve
12	Unsuitable	Obstacle preventing the view of the valve
13	Suitable	Clear view of the entire valve from the front
14	Unsuitable	Cannot see the entire valve, image too close

It is important to note that certain valves that were deemed to have suitable inspection positions prior to incorporating the robot characteristics into the algorithm may no longer have such positions after the modification. Figure 4.5 presents the generated images for valve number 14 in each of the four conducted tests, while Figure 4.6 showcases the results obtained from testing different resolutions, as described in Section 4.2. Notably, the algorithm incorporating the robot characteristics generates inspection positions that are positioned higher, compared to those generated by the 2D search algorithm. As a result, the resulting images may not contain the entire valve within the image.

An overview of the test results achieved by incorporating the robot characteristics into the search algorithm is presented in Table 4.11. The table presents information regarding

Table 4.8: Result from test number two performed on the *huldra-smaller* excerpt of the Huldra facility when incorporating robot characteristics into the search algorithm.

Valve number	Result	Comment
0	Unsuitable	Most of the valve is covered by an obstacle
1	Suitable	Clear view of the entire valve from the front
2	Suitable	Clear view of the entire valve from the front
3	Unsuitable	Image taken of the rear of the valve
4	Suitable	Clear view of the entire valve from the front
5	Unsuitable	Obstacle preventing the view of the valve
6	Unsuitable	Obstacle preventing the view of the valve
7	Unsuitable	Obstacle preventing the view of the valve
8	Unsuitable	Obstacle preventing the view of the valve
9	Unsuitable	Obstacle preventing the view of the valve
10	Unsuitable	Obstacle preventing the view of the valve
11	Unsuitable	Obstacle preventing the view of the valve
12	Suitable	Clear view of the entire valve from the front
13	Suitable	Clear view of the entire valve from the front
14	Unsuitable	Cannot see the entire valve, image too close

Table 4.9: Result from test number three performed on the *huldra-smaller* excerpt of the Huldra facility when incorporating robot characteristics into the search algorithm.

Valve number	Result	Comment
0	Unsuitable	Most of the valve is covered by an obstacle
1	Suitable	Clear view of the entire valve from the front
2	Suitable	Clear view of the entire valve from the front
3	Unsuitable	Image taken of the rear of the valve
4	Suitable	Clear view of the entire valve from the front
5	Unsuitable	Image taken perpendicular to the valve side
6	Unsuitable	Obstacle preventing the view of the valve
7	Unsuitable	Obstacle preventing the view of the valve
8	Unsuitable	Obstacle preventing the view of the valve
9	Unsuitable	Obstacle preventing the view of the valve
10	Unsuitable	Obstacle preventing the view of the valve
11	Unsuitable	Obstacle preventing the view of the valve
12	Unsuitable	Obstacle preventing the view of the valve
13	Suitable	Clear view of the entire valve from the front
14	Unsuitable	Cannot see the entire valve, image too close

the test number, simulation time, and accuracy for each conducted test. In comparison to the outcomes presented in Section 4.2, the incorporation of robot characteristics did not yield an improvement in the accuracy of the algorithm, while resulting in higher run times.

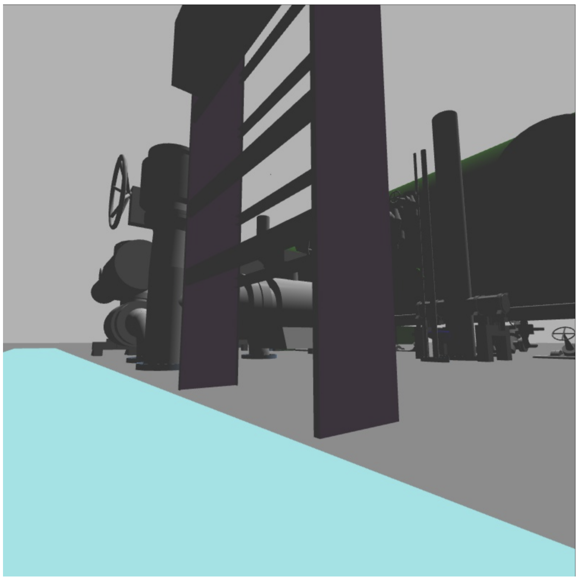
Table 4.10: Result from test number four performed on the *huldra-smaller* excerpt of the Huldra facility when incorporating robot characteristics into the search algorithm.

Valve number	Result	Comment
0	Unsuitable	Obstacle preventing the view of the valve
1	Suitable	Clear view of the entire valve from the front
2	Suitable	Clear view of the entire valve from the front
3	Unsuitable	Image taken of the rear of the valve
4	Unsuitable	Obstacle preventing the view of the valve
5	Unsuitable	Obstacle preventing the view of the valve
6	Unsuitable	Obstacle preventing the view of the valve
7	Unsuitable	Obstacle preventing the view of the valve
8	Adequately suitable	Able to inspect parts of the valve
9	Unsuitable	Obstacle preventing the view of the valve
10	Unsuitable	Most of the valve is covered by an obstacle
11	Suitable	Clear view of the entire valve from the front
12	Suitable	Clear view of the entire valve from the front
13	Suitable	Clear view of the entire valve from the front
14	Unsuitable	Cannot see the entire valve, image too close

Table 4.11: Overview of results when incorporating robot characteristics into the search algorithm.

Test number	Simulation time	Accuracy
1	00:38:40	4/15
2	00:34:38	5/15
3	00:40:56	4/15
4	00:10:03	6/15

(a) Image taken by the robot of valve number 0 when using resolution $5m$. The target valve is located behind the purple object.



(b) Image taken by the robot of valve number 0 when using resolution $1m$. The target valve is located behind the pipe on the right.

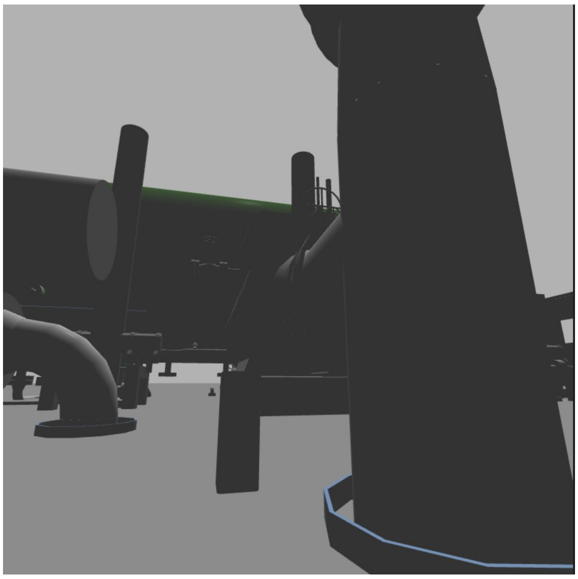
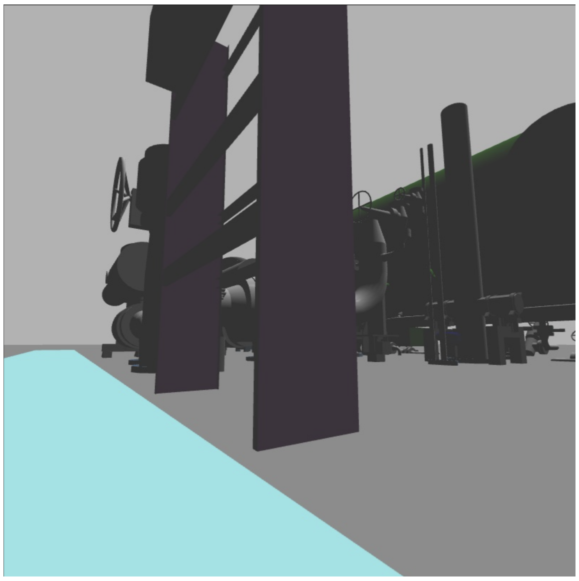


Figure 4.2: Images of valve number 0 when using different resolutions.

(c) Image taken by the robot of valve number 0 when using resolution $0.5m$. The target valve is located behind the purple object.



(d) Image taken by the robot of valve number 0 when using resolution $0.1m$. The target valve is pointed at by the red arrow, located behind the purple object.

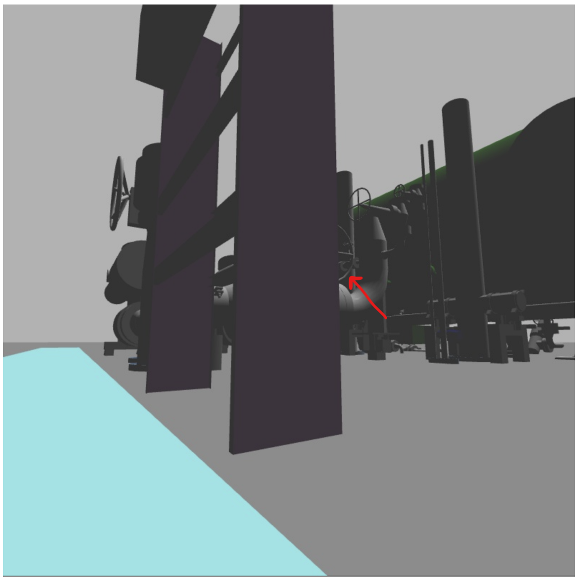
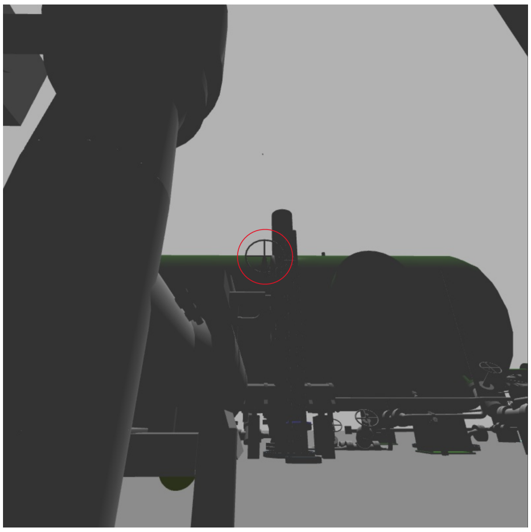


Figure 4.2: Images of valve number 0 when using different resolutions.

(a) Image taken by the robot of valve number 1 when using resolution $5m$. The target valve is seen within the red circle in the middle of the image, viewing the entire valve from the front.



(b) Image taken by the robot of valve number 1 when using resolution $1m$. The target valve is seen within the red circle in the middle of the image, viewing the entire valve from the front.

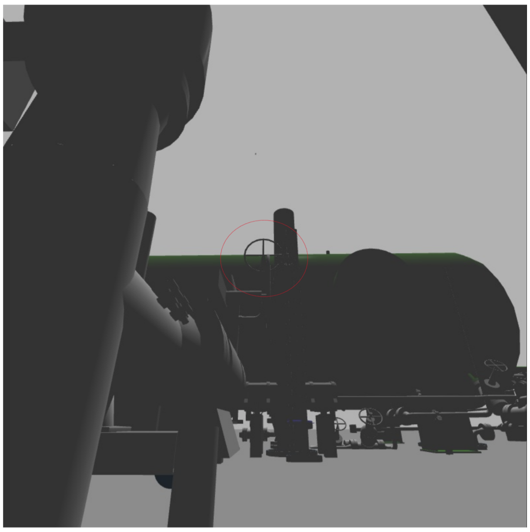
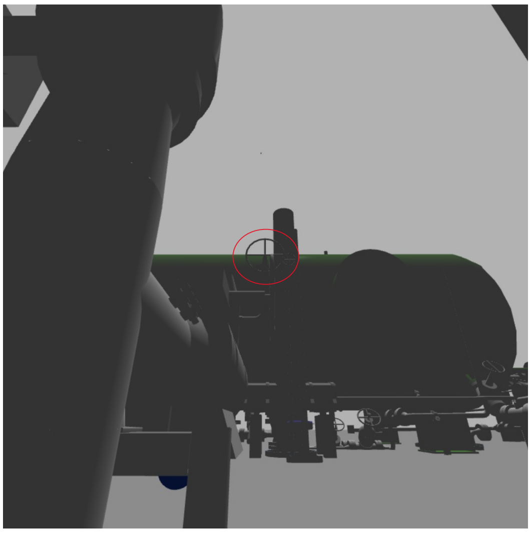


Figure 4.3: Images of valve number 1 when using different resolutions.

(c) Image taken by the robot of valve number 1 when using resolution $0.5m$. The target valve is seen within the red circle in the middle of the image, viewing the entire valve from the front.



(d) Image taken by the robot of valve number 1 when using resolution $0.1m$. The target valve is seen within the red circle in the middle of the image, viewing the entire valve from the front.

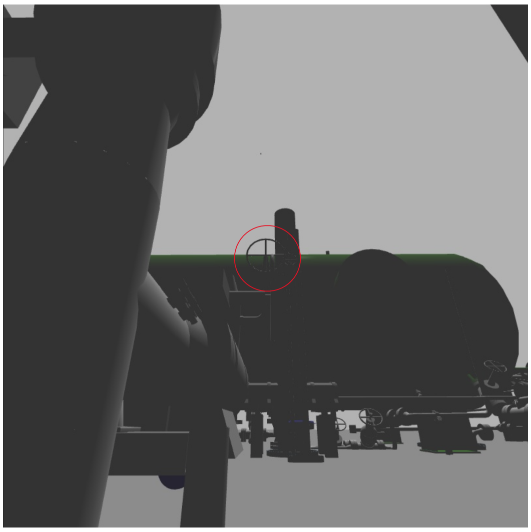
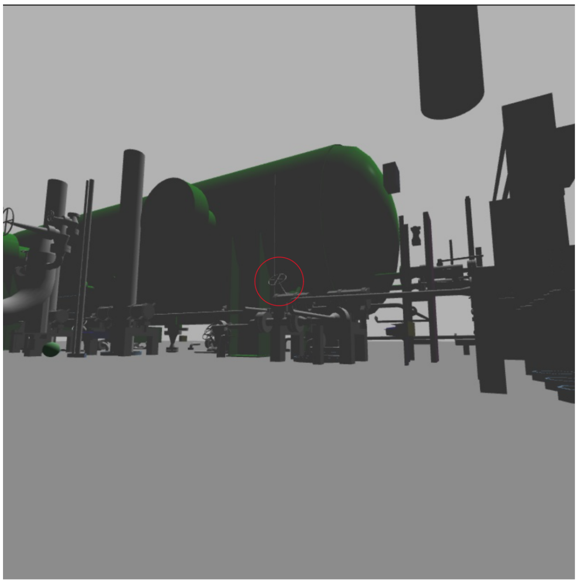
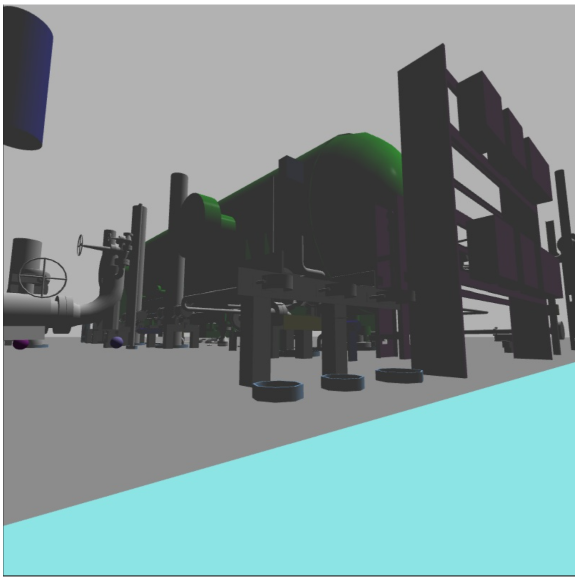


Figure 4.3: Images of valve number 1 when using different resolutions.

(a) Image taken by the robot of valve number 2 when using resolution $5m$. The target valve is seen within the red circle in the middle of the image, viewing the entire valve from the front.

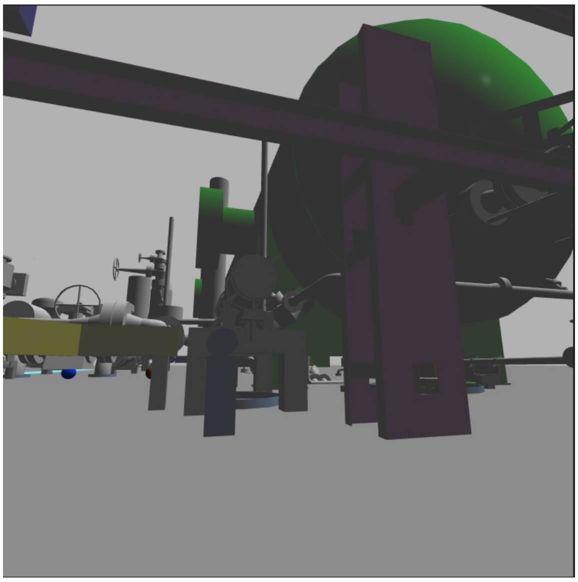


(b) Image taken by the robot of valve number 2 when using resolution $1m$. The target valve is located behind the dark grey object, and cannot be seen in the image.



~~**Figure 4.4:** Images of valve number 2 when using different resolutions.~~

(c) Image taken by the robot of valve number 2 when using resolution $0.5m$. The target valve is located behind the dark grey object, and cannot be seen in the image.



(d) Image taken by the robot of valve number 2 when using resolution $0.1m$. The target valve is seen within the red circle in the middle of the image, viewing the entire valve from the front.

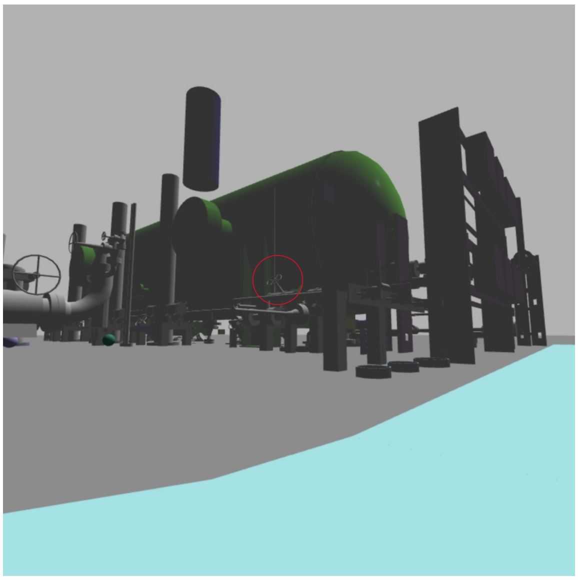
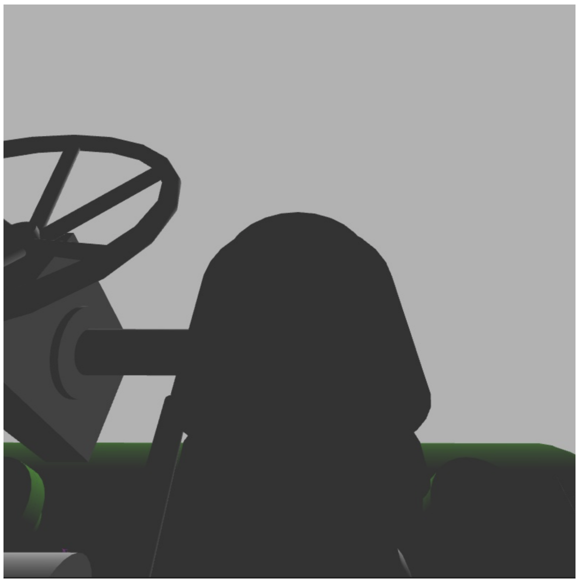


Figure 4.4: Images of valve number 2 when using different resolutions.

(a) Image taken by the robot of valve number 14 when performing test number one. The target valve is located at the left, not able to view the entire valve.



(b) Image taken by the robot of valve number 14 when performing test number two. The target valve is located at the left, not able to view the entire valve.

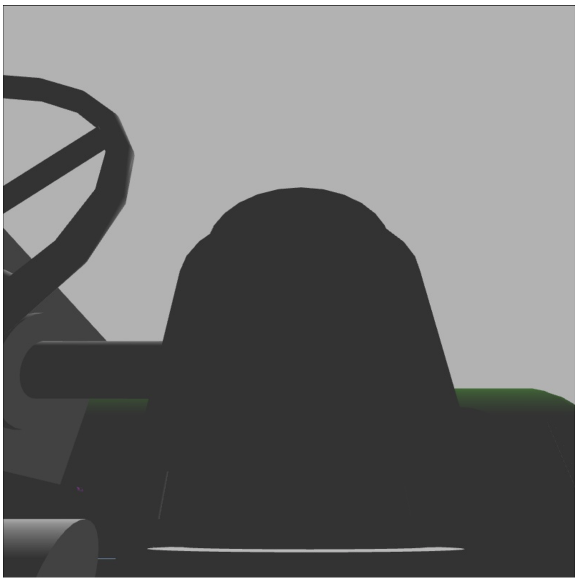
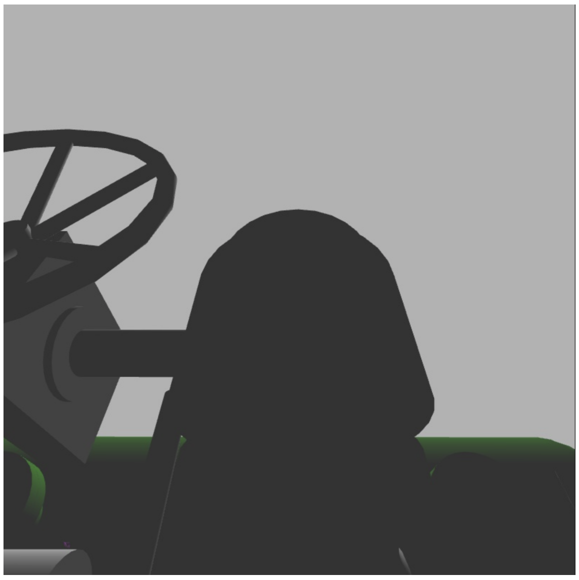


Figure 4.5: Images of valve number 14 when performing tests including the robot characteristics.

(c) Image taken by the robot of valve number 14 when performing test number three. The target valve is located at the left, not able to view the entire valve.



(d) Image taken by the robot of valve number 14 when performing test number four. The target valve is located at the left, not able to view the entire valve.

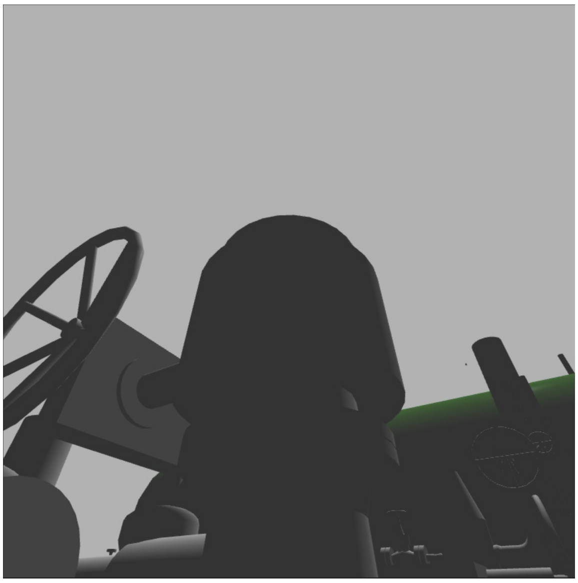
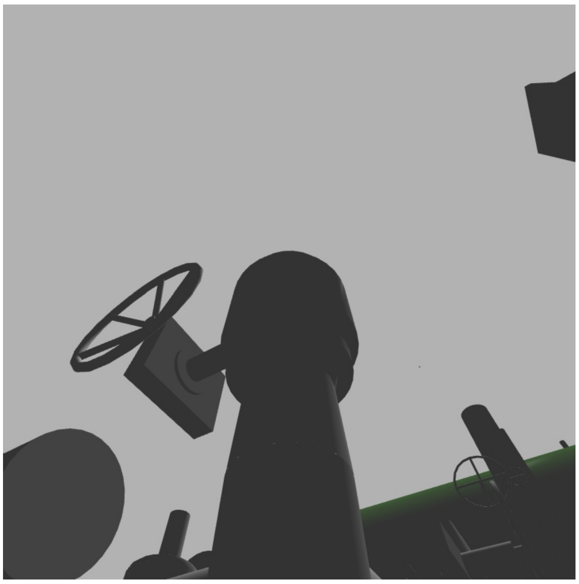


Figure 4.5: Images of valve number 14 when performing tests including the robot characteristics.

(a) Image taken by the robot of valve number 14 when using resolution $5m$. The target valve is located at the center of the image, where one is able to see the entire front of the valve.



(b) Image taken by the robot of valve number 14 when using resolution $1m$. The target valve is located at the center of the image, where one is able to see the entire front of the valve.

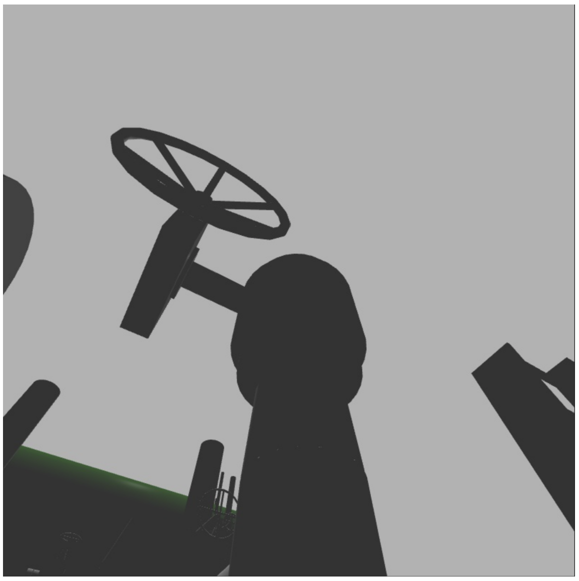
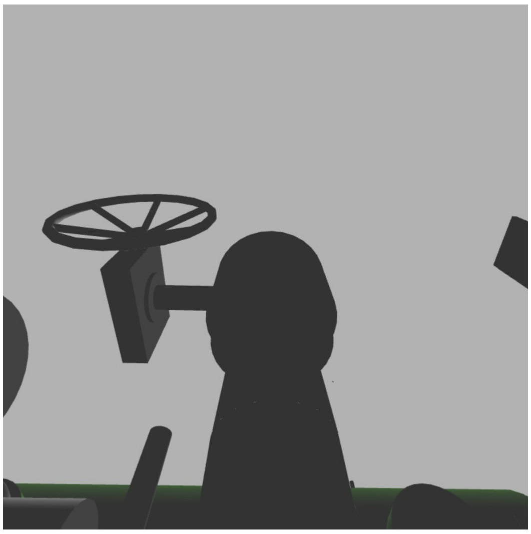


Figure 4.6: Images of valve number 14 when performing a search without the robot characteristics.

(c) Image taken by the robot of valve number 14 when using resolution $0.5m$. The target valve is located at the center of the image, where one is able to see the entire front of the valve.



(d) Image taken by the robot of valve number 14 when using resolution $0.1m$. The target valve is located at the center of the image, where one is able to see the entire front of the valve.

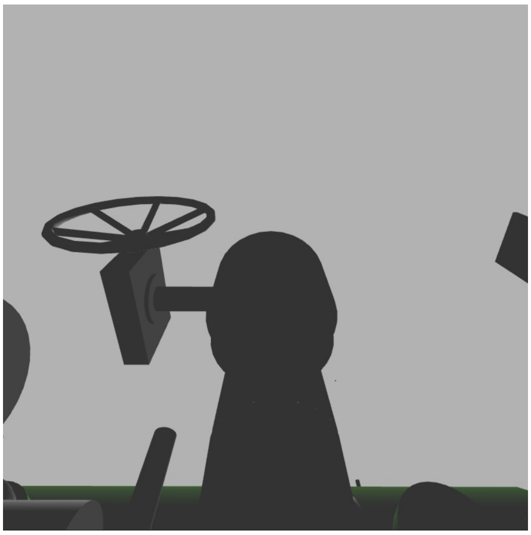


Figure 4.6: Images of valve number 14 when performing tests without the robot characteristics.

5 Discussion

Having presented and analyzed the experimental findings in the preceding sections, this chapter aims to provide a comprehensive interpretation and critical analysis of the results.

5.1 Optimization of the algorithm

The objective of optimizing the algorithm considering both run time and accuracy necessitates making a crucial decision regarding which measure should be prioritized. A time-consuming algorithm undermines the purpose of robotic inspections, rendering them impractical. Simultaneously, the algorithm's accuracy must be satisfactory to avoid the need for manual human inspections. As evident in Table 4.5, increasing the resolution of sampled points along the walkway line amplifies the algorithm's run time. A higher number of samples along the walkway results in more evaluation points for the search algorithm due to reduced variations between each point. Thereby, employing a smaller step length enables the search algorithm to identify additional points during the search, augmenting the dataset of potential inspection points generated by the algorithm. Consequently, more points must be evaluated using the computationally intensive ray tracing scheme. Balancing the trade-off between run time and algorithm accuracy becomes crucial when deploying the algorithm in real-life scenarios, ensuring it remains a valuable asset.

As established in Section 4.2, the search algorithm dramatically reduces the run time compared to the brute force approach, as it limits the evaluation of uninteresting points on the walkway. However, the search algorithm compromises the accuracy of the result. The underlying reason for this compromise lies in the premature termination of the search algorithm for certain valves, leading to the inability to identify clear view inspection positions from the generated candidate inspection positions.

For instance, valve number 0 serves as a noteworthy example where the brute force approach successfully identified suitable inspection positions, whereas the search algorithm failed to do so. The search algorithm initiates the search process from the walkway point with the minimum distance to the target valve and progressively explores directions aligned with the valve's orientation. As depicted in Figure 4.2, it becomes apparent that the search algorithm terminates prematurely based on the angle between the camera direction and the optimal inspection direction of the valve, rather than achieving the optimal angle. This observation suggests the need to revise the termination criterion of the search algorithm or employ an alternative metric to determine the starting point of the search.

The algorithm can be terminated based on various criteria, such as setting a threshold on the score of the inspection positions found, limiting the number of iterations, or employing computer vision techniques for the detection of interesting qualities of the valves to be inspected. With the integration of computer vision, the algorithm can automatically inspect the valve during run-time, removing the need for manual inspection of produced images. The search can be terminated once an appropriate position is discovered where the entire front of the valve is visible and can be effectively inspected.

An alternative approach for finding the start point of the search involves disregarding distance as a primary metric and instead utilizing a camera equipped with zooming capabilities. By doing so, the significance of the distance measure is diminished. In this scenario, the angle between the optimal inspection direction and the camera direction can serve as the basis for the starting point, where one may start the search where the angle between the camera direction and the optimal inspection direction is 0 degrees. The search can then proceed outwards from this starting point, rather than inwards towards the optimal inspection angle of zero degrees.

Another underlying reason for the search algorithm not being able to identify suitable inspection positions for all valves is that it relies heavily on the orientation of the target valve to determine the next valid search point along the walkway. The algorithm terminates the search when the absolute value of the angle between the camera direction and the optimal inspection direction of the target valve no longer decreases. Figure 5.1 provides an illustrative example of the iterative process employed by the search algorithm for valve number 0. In iteration 0, the algorithm initiates by determining the starting point through the exploitation of the minimum distance between the target valve and the walkway points. Subsequently, in iterations $1, 2, 3$ and 4, the algorithm calculates and projects potential inspection points onto the walkway. However, in iteration 4, the projected point on the walkway coincides with that of iteration 3. Consequently, the angle between the camera direction and the valve's optimal inspection direction remains unchanged in both cases, causing the search algorithm to terminate.

The reliance on the orientations for finding the next search point is also particularly challenging for valves with orientations perpendicular to the ground plane. In such cases, when the start point of the search is rotated, the projected point on the walkway remains the same. As a result, relying solely on the valve's orientation does not provide adequate information about the direction in which to move the start point. Consequently, there is a limited number of potential inspection positions, leading to early termination of the search for these valves. For instance, in Section 4.2, it is demonstrated that the algorithm fails to yield suitable inspection positions for valve number 5, which exhibits a perpendicular orientation relative to the ground floor. Figure 5.2 illustrates the search iterations for valve number 5, where iteration 0 and iteration 1 yield the same point, thereby terminating the search process. In this specific case, only the starting point is evaluated, determined based on the minimum distance between the target valve and the walkway line, as well as the points defined by the height interval when incorporating the robot characteristics. The

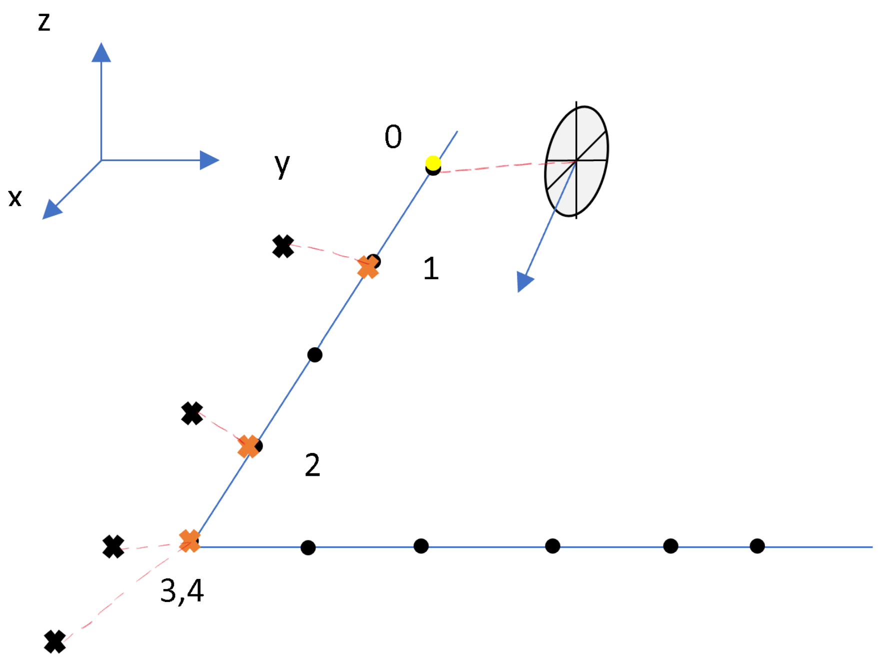


Figure 5.1: Illustration of how the search algorithm works on valve number 0. The valve is illustrated by the wheel, with a corresponding arrow showing its orientation. The black circles illustrate the samples along the walkway line. Each iteration of the algorithm has a corresponding iteration number, from 0 to 4. The yellow circle illustrates the starting point of the search found by the minimum distance between the target valve and the sampled points along the walkway. The black crosses illustrate the next iteration point before projecting onto the walkway, while the orange crosses show the iterations after projecting onto the walkway. The red dotted lines illustrate the minimum distances from the possible inspection points to the projected point on the walkway.

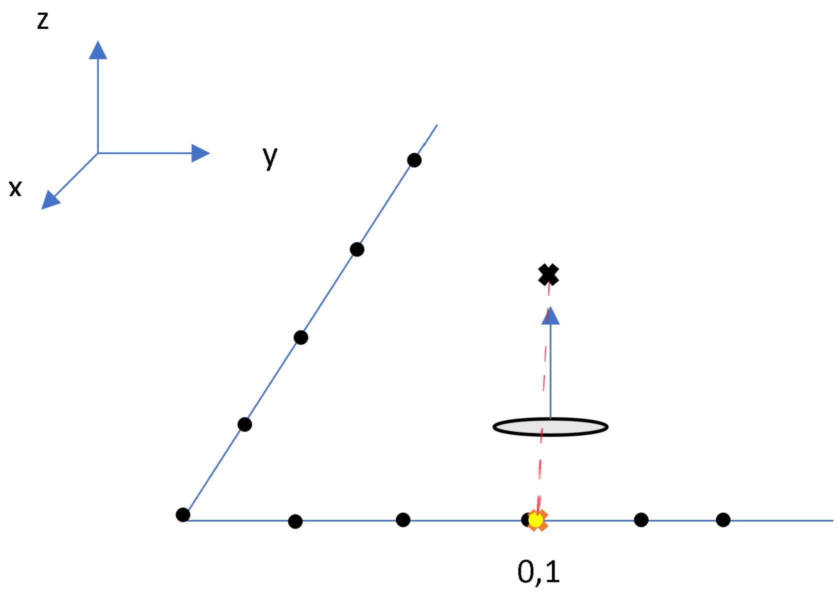


Figure 5.2: Illustration of how the search algorithm works on valve number 5. The valve is illustrated by the oval, with a corresponding arrow showing its orientation. The black circles illustrate the samples along the walkway line. Each iteration of the algorithm has a corresponding iteration number, from 0 to 1. The yellow circle illustrates the starting point of the search found by the minimum distance between the target valve and the sampled points along the walkway. The black cross illustrates the next iteration point before projecting onto the walkway, while the orange cross shows the iteration after projecting onto the walkway. The red dotted line illustrates the minimum distances from the possible inspection points to the projected point on the walkway.

suboptimal performance of the algorithm can thus be attributed to the aforementioned issue, whereby the projection of subsequent search points onto the walkway based solely on the valve's orientation leads to repetitive convergence onto the same point within the constrained solution space.

Furthermore, it is important to acknowledge that the algorithm employs a local optimization strategy, operating within a particular region of the search space. Consequently, the resulting position obtained from the algorithm may not necessarily represent the global optimum. The local optimization technique is executed only once per valve, generating the same starting position (i.e., the point on the walkway with the shortest distance to the target valve) for each algorithm run. However, it is possible that the local optimum may produce a suboptimal outcome compared to an alternative starting point, even if the latter has a slightly greater distance from the inspection point to the target valve. By implementing a repeated search for each target valve before determining the optimal inspection position, it is possible to achieve a greater number of suitable inspection points. This approach may also help mitigate the limitations associated with the search algorithm's heavy reliance on the orientation of the target valves. Through repeated searches, alternative inspection positions can be explored and evaluated, potentially leading to improved results and overcoming the limitations imposed by the orientation-based approach.

When integrating the robot characteristics into the search algorithm, more points are subject to evaluation through ray tracing. This occurs because each potential point identified by the search algorithm in the search process undergoes additional sampling of possible search points in the vertical direction and subsequent ray tracing, resulting in increased run time. To limit the increase in run time, it is possible to decrease the number of points requiring the ray tracing scheme for clear visibility evaluation. One approach to decrease the number of ray tracing evaluations involves considering the height interval only for the point deemed optimal by the search algorithm without the integration of robot characteristics. This approach significantly reduces the number of potential inspection points. However, it should be noted that finding the optimal inspection point first and then exploring the vertical direction for the height interval might result in the algorithm missing a suitable point if the wrong height is selected before sampling of the height interval and detecting suitable points in the vertical direction. The trade-off between run time and accuracy is again important, and it is crucial to ensure that optimization efforts do not compromise the algorithm's accuracy. During the implementation process, it was determined that the additional computational effort associated with sampling the height interval for all possible search points per valve was warranted.

The analysis of the results also reveals that integrating robot characteristics in the vertical direction within the search process does not lead to more appropriate inspection positions. In fact, it yields the opposite effect. The algorithm encounters challenges when identifying valves near the walkway line. The generated inspection positions tend to yield higher scores for positions located above the ground floor, resulting in the captured images being too close to the target valve to be fully contained within the image frame. This limitation hinders the algorithm's ability to capture the necessary visual information for accurate inspection effectively. The assumption that shorter distances result in superior inspections is contradicted in this context, as capturing the entire valve necessitates taking the image from a certain distance. Consequently, a remedy to address this issue would involve implementing a threshold on the lower end of the distance measure. This threshold would ensure that the generated images are captured at an appropriate distance from the target valve, enabling the comprehensive coverage of the valve within the image frame. By incorporating such a threshold, the algorithm can effectively manage the distance constraint and ensure the production of satisfactory inspection results.

Furthermore, while incorporating robot characteristics in the vertical direction, where the camera is mounted on a robotic arm, allows for views from different heights along the walkway, it does not address the issue of valves that are challenging to observe from the walkway itself. To overcome this limitation, the algorithm could benefit from the inclusion of a robotic arm capable of horizontal movement beyond the walkway line. By introducing an additional degree of freedom in the horizontal direction, along with the existing vertical degree of freedom, more valves within the facility could be identified, possibly leading to improved algorithm accuracy.

Another significant limitation of the proposed algorithm including robot characteristics, is the lack of guarantee that a robot meeting the required characteristics for conducting the inspection from the generated inspection positions exists or is readily available. To ensure the algorithm's practical applicability in real-world scenarios, it may be necessary to impose constraints on the robot characteristics provided to the algorithm. These restrictions would ensure that the optimal inspection positions generated by the algorithm are feasible and reachable for the specific robot employed.

Moreover, the algorithm does not consider the size or shape of the robot being utilized. The chosen inspection robot may have limitations in terms of its dimensions and maneuverability within the facility. Consequently, the generated inspection points may be located in areas where there is insufficient space for the robot to physically reach the desired inspection position. It is essential to incorporate constraints related to the robot's size and dimensions requirements into the algorithm to ensure that the generated points are feasible and compatible with the specific robot being employed.

5.2 Further work

The developed algorithm highlights the significant potential for employing robot based inspections in industrial facilities, thereby replacing human performed inspections. While it is acknowledged that the algorithm does not succeed in identifying suitable inspection positions for all valves within the testing facility, the results demonstrate the effectiveness of the proposed search algorithm in reducing computational time through intelligent search strategies.

However, it is important to recognize that the algorithm still has certain limitations that is necessary to handle in order to bring the algorithm to an acceptable level of accuracy and run time to be applicable in real life scenarios. Based on the discussion in the previous section, there have been identified several recommendations for future work.

The need to enhance the accuracy of the search algorithm becomes crucial given the decrease in accuracy observed when transitioning from the brute force approach. This objective can be pursued through various approaches, as highlighted in the discussion. One particularly effective method involves incorporating a repetitive search strategy, employing different starting points to improve the resilience of the local optimization technique employed for identifying the optimal inspection positions.

Since the search algorithm already calculates the distances between the target valve and the walkway points, an opportunity arises to utilize additional points located within a specified distance threshold as alternative starting points. By repeating the search process with multiple starting points, the optimal inspection point is determined for each search iteration, and subsequently, the best inspection point is selected from the optimal inspection points obtained through different searches. This selected point can then be regarded as the ultimate optimal inspection point.

Another challenge highlighted in both this Master's thesis and the specialization project [1] is the difficulty of locating suitable inspection positions for valves that are hardly visible from the walkway. Consequently, it is recommended to incorporate robot characteristics not only in the vertical direction but also in the horizontal direction, as this may enable the inspection of other valves from such positions.

Furthermore, the images captured by the inspector robot do not readily indicate which valve is the target valve in each image. Hence, it would be worthwhile to explore the possibility of automatically detecting the target valve within the generated images. Although the algorithm possesses information about the specific valve serving as the target in each image, manual inspection of the images can be challenging when identifying the actual target valve. The simplest approach to address this issue is to incorporate a form of highlighting for the target valve in the virtual environment before the robot captures the image. However, in the long run, leveraging computer vision techniques for automatic image detection could prove beneficial in fully harnessing the potential of robot-based inspection methodologies, thereby replacing the need for manual inspection of each image.

6 Conclusion

In conclusion, this study aimed to further develop an algorithm for computing optimal inspection positions for inspecting valves in industrial facilities, based on the findings in the specialization project [1]. To achieve this, several sub-objectives were established. These included the integration of the planning algorithm into Equinor's system, adapting the optimization algorithm to ensure scalability in terms of run time for larger facilities, considering a movable camera position in optimizing inspection positions, and providing recommendations for future research.

The developed algorithm presented in this study offers a notable reduction in run time compared to the brute force approach of evaluating all potential inspection points along the walkway. However, the algorithm exhibits significant limitations in terms of accuracy. As a result, this thesis proposes incorporating a repetitive search methodology with distinct starting positions for each iteration to enhance the algorithm's performance.

Furthermore, the thesis establishes that integrating robot characteristics solely in the vertical direction, without considering characteristics in the horizontal direction, does not yield more suitable inspection positions. To address this, it is recommended to incorporate robot characteristics in both the vertical and horizontal directions. This enhancement could enable the inspection of valves that are not fully visible from the walkway while simultaneously improving the overall results for valves that are visible from the walkway.

In summary, this study has provided valuable insights into the theoretical foundations and practical implications of implementing an algorithm for robotic inspections. Moreover, it has put forward several propositions to refine the algorithm's applicability to real-world scenarios.

Bibliography

[1] Åshild Berg Rosland. Robot based inspection of valves and sensors in industrial facilities, 2022.

[2] Chunguang Bai, Patrick Dallasega, Guido Orzes, and Joseph Sarkis. Industry 4.0 technologies assessment: A sustainability perspective. *International Journal of Production Economics*, 229:107776, 2020.

[3] Tayaba Abbasi, King Hann Lim, Toufique Ahmed Soomro, Idris Ismail, and Ahmed Ali. Condition based maintenance of oil and gas equipment: A review. In *2020 3rd International Conference on Computing, Mathematics and Engineering Technologies (iCoMET)*, pages 1–9, 2020.

[4] Dharminder Singh, Matthew Charlton, Taimoor Asim, and Rakesh Mishra. Design for additive manufacturing and its effect on the performance characteristics of a control valve trim. *International Journal of COMADEM*, 22(2):59–67, August 2019.

[5] National Institute for Occupational Safety and Health. Niosh-osha hazard alert. health and safety risks for workers involved in manual tank gauging and sampling at oil and gas extraction sites. https://www.osha.gov/sites/default/files/publications/OSHA3843.pdf, 2016. 18-12-2022.

[6] Erlend Blomseth. Planlegging av inspeksjoner på industrielle anlegg ved hjelp av digital tvilling, 2022.

[7] Equinor ASA. Equinor's strategy. https://www.equinor.com/about-us/strategy. Visited 28-02-2023.

[8] E. Blomseth. Deteksjon og inspeksjon av ventiler på industrielle anlegg, 2021.

[9] C. Day A. Fuller, Z. Fan and C. Barlow. *Digital twin: Enabling Technologies, Challenges and Open Research*. IEEE, 2020.

[10] Adil Rasheed, Omer San, and Trond Kvamsdal. Digital twin: Values, challenges and enablers from a modeling perspective. *IEEE Access*, 8:21980–22012, 2020.

[11] Siemens Digital Industries Software. 3d-modeling. ttps://www.plm.automation.siemens.com/global/en/our-story/glossary/3d-modeling/17977. Visited 30-11-2022.

[12] K. McHenry and P. Bajcsy. *An Overview of 3D Data Content, File Formats and Viewers*. National Center for Supercomputing Applications, 2008.

[13] M. Dawson-Haggerty. trimesh. https://trimsh.org/trimesh.html. 08-12-2022.

[14] J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer New York, NY, 2006.

[15] X. Yang. *Nature-Inspired Optimization Algorithms*. Elseviev, 2014.

[16] Stuart J. Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach (Fourth Edition)*. Pearson Education, 2022.

[17] Jongchan Baek, Hayeong Jeon, Gwangjin Kim, and Soohee Han. Visualizing quaternion multiplication. *IEEE Access*, 5:8948–8955, 2017.

[18] D. Rose. Rotation quaternions, and how to use them. https://danceswithcode.net/engineeringnotes/quaternions/quaternions.html, 2015. Visited 14-05-2023.

[19] H. Durrant-Whyte and T. Bailey. Simultaneous localization and mapping: part i. *IEEE Robotics & Automation Magazine*, 13(2):99–110, 2006.

[20] Bashar Alsadik and Samer Karam. The simultaneous localization and mapping (slam)-an overview. *Journal of Applied Science and Technology Trends*, pages 120–131, 2021.

[21] Robotis. Turtlebot3. https://emanual.robotis.com/docs/en/platform/turtlebot3/overview/. Visited 06-03-2022.

[22] Robotis. Features. https://emanual.robotis.com/docs/en/platform/turtlebot3/features/. Visited 15-05-2023.

[23] Docker Inc. Docker overview. https://docs.docker.com/get-started/overview/. Visited 24-11-2022.

[24] The ros ecosystem. https://www.ros.org/blog/ecosystem/. Visited 22-11-2022.

[25] The Robotics Back-End. What is ros? https://roboticsbackend.com/what-is-ros/. Visited 19-05-2023.

[26] Gazebosim. https://gazebosim.org/home. Visited 22-11-2022.

[27] Rviz. http://wiki.ros.org/rviz. Visited on 22-11-2022.

[28] Equinor. Isar. https://github.com/equinor/isar. Visited 08-03-2023.

[29] Equinor. isar-turtlebot. https://github.com/equinor/isar-turtlebot. Visited 08-03-2023.

[30] Addisu Taddese. Scustom elements and attributes. http://sdformat.org/tutorials?tut=custom_elements_attributes_proposal. Visited 08-05-2023.

[31] Robotis. Turtlebot3 slam simulation. https://emanual.robotis.com/docs/en/platform/turtlebot3/slam_simulation/. 09-03-2023.

[32] Christoph Gohlke. transformations. http://docs.ros.org/en/jade/api/tf/html/python/transformations.html. Visited 21-04-2023.

[33] Inc. pandas via NumFOCUS. About pandas. https://pandas.pydata.org/about/. Visited 22-05-2023.

[34] NumPy Developers. numpy.linspace. https://numpy.org/doc/stable/reference/generated/numpy.linspace.html. Visited 08-05-2023.

Appendix

A Valve overview

Table 6.1: Overview of the valves in the Huldra-smaller model.

Valve number	Valve tag
0	20-2000VF
1	20-2007VF
2	20-2003VF
3	20-2006PL
4	20-2001PL
5	20-2031PL
6	20-2001WI
7	20-2002WI
8	20-2003WI
9	20-2004WI
10	20-2005WI
11	20-2006WI
12	20-2001VF
13	20-2008VF
14	43-4507VF

B Valve visibility overview

Table 6.2: Overview of the visibility of the valves in the excerpt of the Huldra model used in testing.

Valve number	Visibility from walkway
0	Fully visible from walkway
1	Fully visible from walkway
2	Fully visible from walkway
3	Fully visible from walkway
4	Fully visible from walkway
5	Hardly visible from walkway
6	Hardly visible from walkway
7	Hardly visible from walkway
8	Hardly visible from walkway
9	Visible from walkway
10	Fully visible from walkway
11	Fully visible from walkway
12	Hardly visible from walkway
13	Fully visible from walkway
14	Fully visible from walkway

C Illustration of the excerpt of the Huldra facility's digital twin

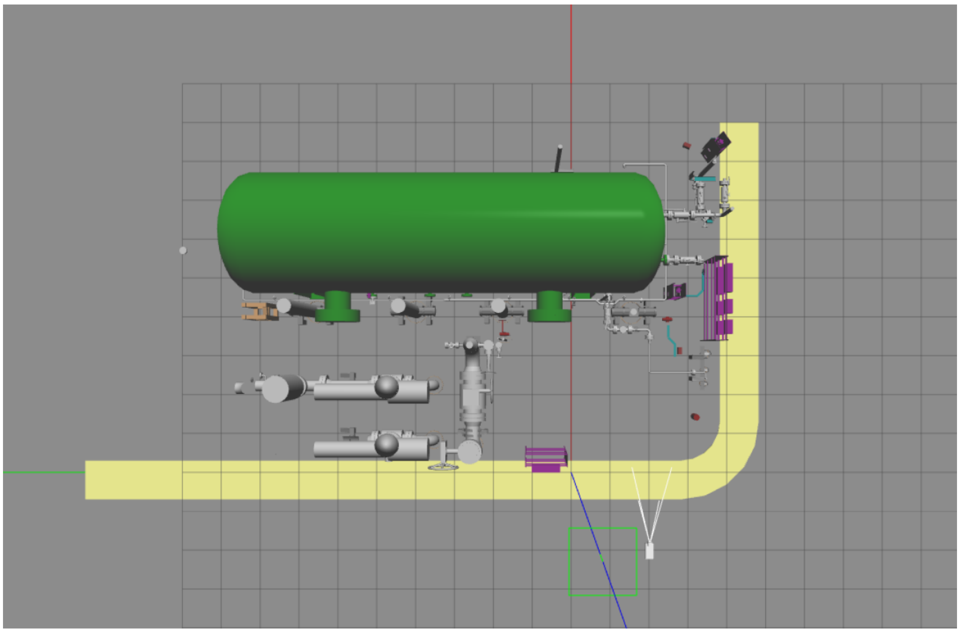


Figure 6.1: Illustration of the digital twin of the excerpt of the Huldra facility referred to as *huldra-smaller*.