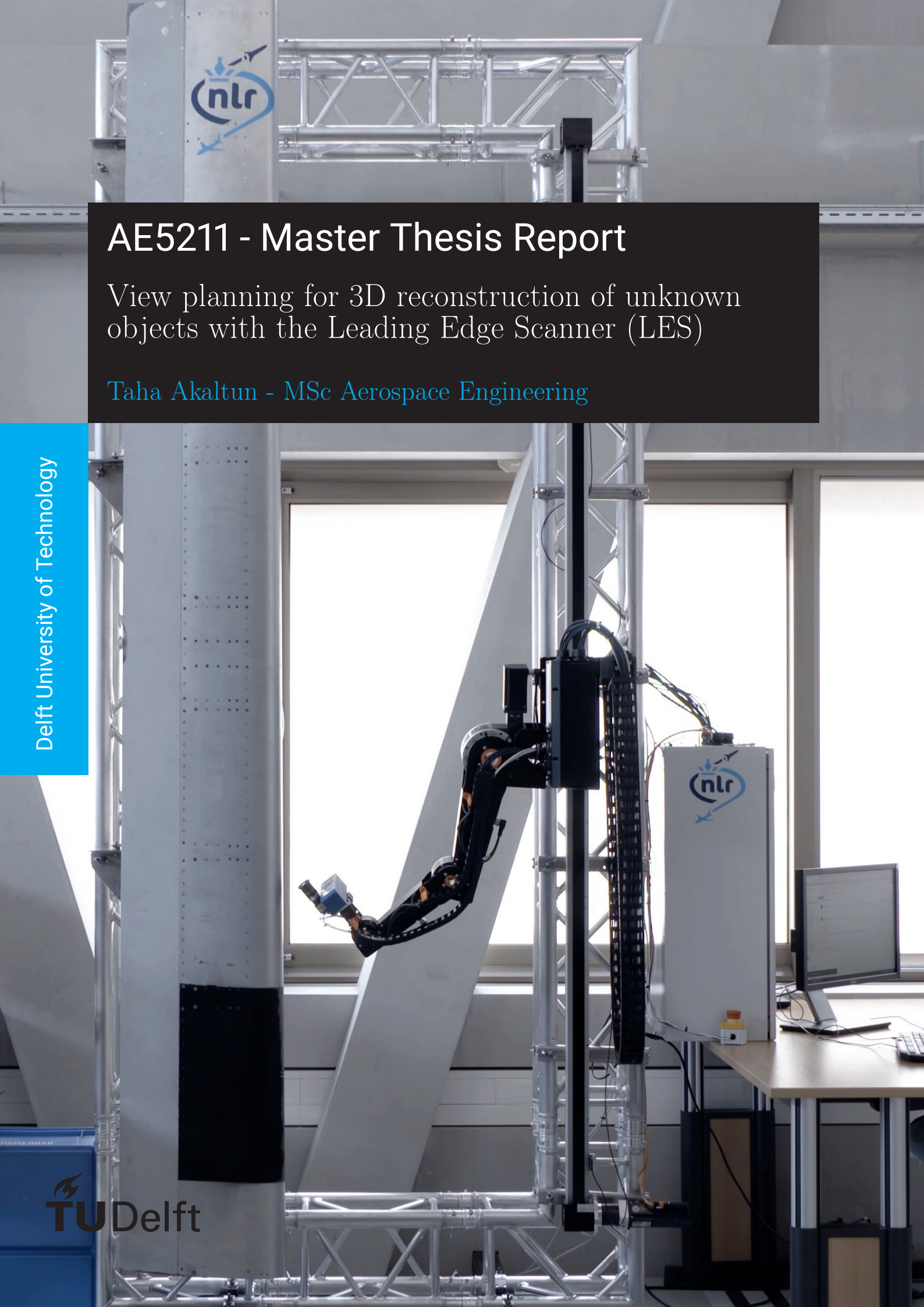


AE5211 - Master Thesis Report

View planning for 3D reconstruction of unknown objects with the Leading Edge Scanner (LES)

Taha Akaltun - MSc Aerospace Engineering

Delft University of Technology

TU Delft

AE5211 - Master Thesis Report

View planning for 3D reconstruction of unknown objects with the Leading Edge Scanner (LES)

by

Taha Akaltun - MSc Aerospace Engineering

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on December 7, 2021 at 10:00 AM.

Student number:	4007263	
Project duration:	February 15, 2021 – November 15, 2021	
Thesis committee:	Dr. R. Groves,	TU Delft, Supervisor, Chair
	Dr. K. Masania,	TU Delft, Examiner
	Dr. N. Eskue,	TU Delft, Committee member
	Dr. F. Oliveiro,	TU Delft, Committee member (FPP)
	MSc. D. Spirtovic,	NLR, Supervisor

An electronic version of this thesis is available at http://repository.tudelft.nl/.

TU Delft

Contents

List of Figures

List of Tables

List of Acronyms

List of Symbols

1
Executive Summary

Automation has come knocking at the door of the world of aircraft maintenance. The general trend shows that the total aircraft fleet size of the world increases in contrast to the number of technicians, which decreases. With the outbreak of the coronavirus disease more technicians left the sector, further triggering the demand of automation of frequent aircraft inspections. Therefore, the Royal Netherlands Aerospace Center (NLR) launched a project and setup called Leading Edge Scanner (LES) to automate technician performed inspections on aircraft for partial or total replacement of this service. The LES will automatically detect and classify anomalies on aircraft components.

Automated three-dimensional (3D) reconstruction of known or unknown objects requires robots and sensors. As several images from different viewpoints are needed to reconstruct a 3D computerized object, a view planning strategy is required to find the Next-Best-View (NBV). This is necessary to automate the view planning process for where to look next after an initial scan is taken of an object for reconstruction and further processing. The sensor system needs to be located at the desired location. Afterwards, raw images must be processed to eventually come to a fully reconstructed 3D model of the inspected object. The challenging problem of finding the NBV has been studied since the 1980s.

The LES is a five Degrees of Freedom (DOF) system. It consists of a robotic arm with two sensors attached to its end-effector. The sensor payload consists of a monochrome stereoscopic 3D sensor and a two-dimensional (2D) Red Green Blue (RGB) sensor. The robotic arm is placed on a linear slider. This system can position the sensors to a desired location and thereby viewpoint, which is relative to the object. The 3D sensor produces a point cloud, which after post-processing and scans from different poses is stitched together to obtain the 3D model of an object. Other general data structures are the triangle mesh and voxels. The reconstruction workflow consists of robot positioning, scanning, registration and update and planning the NBV. This process continues until a termination criterion is met.

This report explains the theory behind NBV, compares NBV algorithms and strategies and simulates the scanning of several objects with the constraints of LES. The research question is about which NBV algorithm is applicable to the LES to be used in real scenarios. Answers on the sub questions how the imaging pipeline is set up for image acquisition and object registration (1), which kind of sensor systems are required for unknown object reconstruction (2) and which prominent view planning algorithms are available (3) have lead to the final answer to the presented research question.

The principles of the algorithm Surface Edge Explorer (SEE) is used for this simulation. It is surface based, uses point clouds and their boundaries for NBV calculation. It outperforms volumetric approaches in more surface coverage in less number of scans. Volumetric methods also come with relatively more computational costs due to large memory usage.

Six NBV selection methods are simulated in order to find the most optimum NBV strategy for LES. The hybrid option, which is selecting the candidate points with the minimum distance to the sensor and minimum curvature resulted in full coverage of the scanned model. The proposed NBV algorithm for the LES is the SEE with the hybrid NBV selection method. The basis of SEE is border detection and normal vector calculation. The hybrid method uses the ranking method of minimum distance and minimum curvature combined and ensures full surface coverage at an acceptable computational time and total sensor movement distance. It also terminates more stable than the other only distance and curvature methods, which most of them not being capable of realising full surface coverage.

2 Introduction

In order to automatically reconstruct a three-dimensional (3D) model of unknown objects, a view strategy or planning is required. Automatically finding the next viewpoints, called the Next-Best-View (NBV), can partially or totally replace the manual work that is needed during scanning of objects with sensors, eventually reducing workloads, costs and time. Another possible benefit is the improvement of the model quality.

The Royal Netherlands Aerospace Center (NLR) intends to automate visual inspections performed on aircraft. To reach this goal, several research projects are being conducted, one of them being for object reconstruction in a computerized 3D environment. This reconstruction needs a sensor system, which acquires a set of multiple images, and a positioning system to position the sensors.

Another NLR research project led to the current experimental setup called the Leading Edge Scanner (LES): a robotic arm with two sensors, mounted on the end-effector. This five Degrees of Freedom (DOF) positioning system also consists of a linear module to move the robotic arm with attached sensors. It can position the sensor system to a desired location and thereby viewpoint, which is relative to the object.

Multiple images from different viewpoints are necessary to reconstruct an object in 3D. These images are merged to form the desired 3D model. These images are taken by sensors, which collect depth data of point clouds from various viewpoints. Currently, the image acquisition is done manually where the LES requires viewpoint coordinates for moving the sensors to the desired location.

As part of the desired automation and quality, the LES needs an algorithm to predict the next camera pose without human intervention after starting the reconstruction process. The latter will be the scope of this thesis assignment, in this case for unknown objects. The so called view planning algorithm, or NBV, is about finding the optimal view sequence to automatically reconstruct an object by focusing on the best model quality and fewer robot movements. Beside the desired automation, this improves the efficiency of the system in terms of runtime, quality and computational costs.

The ultimate goal of the LES is to automate visual anomaly detection and classification to increase damage assessment on the leading edge of wings. This requires the automation of image acquisition. The project stage of view planning for 3D reconstruction of unknown objects is reached after previous research projects at NLR. These were on the field of investigating hardware and software for robotic solutions, hardware and sensor systems for inspection purposes, software and algorithms for 3D image acquisition and processing like noise filtering, downsampling and merging of images. Also research on software and algorithms for two-dimensional (2D) and 3D defect detection and classification is currently researched.

The area of interests of this academic research are the theory and the reasoning behind NBV algorithms, including relevant constraints and requirements. The output of this research is a MATLAB model describing the most applicable NBV algorithm for the LES. The simulation visualizes the calculated new viewpoints and the results of the NBV algorithm being the surface coverage, total number of scans, computation time and distance traveled by the robotic arm. The chosen algorithm will be used by the LES to further automate the system.

The NBV strategy is based on boundary detection, data segmentation, normal and curvature calculations and NBV ranking. As many new sensor positions are possible, ranking of NBV's is a crucial step. The chosen method and algorithm needs to self-terminate. Several methods and objects are simulated and compared to arrive at an applicable strategy for LES. These are the minimum distance, maximum distance, the latter two combined, minimum curvature(s) and a hybrid method which is the minimum distance and minimum curvature combined. The simulated objects are a sphere, cone, cube and plate, which are represented by a point cloud.

Chapter 3 describes the hardware required for object reconstruction. Chapter 4 contains the theory behind NBV and a comparison of common NBV algorithms. Chapter 5 presents the NBV strategy, ranking and simulation. The results are given and discussed in Chapter 6. Chapter 7 contains recommendations for future work. The final Chapter, 8, concludes this thesis. Appendices A, B and C contain the codes of the complete simulation. Appendix D presents the NBV results of the simulated objects.

3
Hardware: Sensors and Robots

This Chapter starts with an introduction to automated 3D object reconstruction (3.1) and describes the types of sensors (3.2) used. Section 3.3 is about robots used for sensor positioning. Section 3.4 presents the available experimental setup at NLR, the LES. The summary can be found in Section 3.5.

3.1. Introduction to automated 3D object reconstruction

3D object reconstruction requires scans from multiple viewpoints of an object. A single view can contain insufficient information or inadequate coordinate data about the scanned object. This leads to the necessity of planning robot movement based on visual information and algorithms. View planning is crucial to achieve specific goals in the best way, in this case to be able to predict the best next view pose and also to reach that position. Then it is required to reconstruct an object in the most optimal manner considering model quality, runtime and computation costs. Active vision tasks are generally fulfilled by sensors and robots [1].

3.2. Sensors

Cameras for shape measurement or visual sensing, called sensors, are mainly divided in two groups, being passive or active devices [2]. These type of sensors obtain 2.5D information about a scanned object and calculate the depth and thereby the distance of the scanned object surface in view to the sensing system.

Passive range sensors need an external light source and are thereby independent of the hardware [1]. Sensing of passive sensors is based on stereo vision, obtaining depth information just like the human eye. According to Zeng et al., passive measuring has two main disadvantages. Firstly, the scanned object needs to have texture on it to be able to obtain qualitative data. The second disadvantage is the reduced measurement performance when changes in lighting occurs. Active sensors use their own light source and therefore are less affected by external light factors. Time-of-Flight (ToF) and structured light sensors are active sensors and obtain depth information by projecting light onto an object. The reflected light is captured and used for calculating the distance [1].

Zeng et al. compared the three above described widely used measurement techniques and types of range sensors [1]. Table 3.1 shows the compared three sensor types based on distance performance, accuracy, influence of light, depth measurement, resolution and cost.

It is important to note that all scanners have their optimal working distance [3]. A sensor can not be placed too far to the object, nor too close. The Object Bounding Box (OBB) is related to the furthest working distance of the sensor.

Table 3.1: *Comparison of depth sensors [1]*

Type	Technique	Distance	Accuracy	Influence of light	Depth	Resolution	Cost
Stereo vision	Passive	Short	High	Strong	Sparse	High	Low
ToF	Active	Long	Low	Little	Dense	Low	High
Structured light	Active	Short	High	Little	Dense	Low	High

Zeng et al. states that, for relatively short distances to the object, stereo vision [4] can be used. In the project of NLR and considering the current set-up, the scanning of aircraft components take place at relatively short distances, varying from 0.5 m to 1.5 m. Stereo vision has high accuracy and is strongly affected by a change in lighting conditions. Zeng et al. conclude that the depth measurement is sparse, meaning that the obtained point density is lower than the other ones and that data points are relatively further away from each other. The resolution is high at low cost.

The ToF sensor [2], using a laser, is used for medium to long ranges, as stated by Zeng et al. via Scott and Roth. As for the active sensors, ToF can have low accuracy. This is the opposite for structured light sensors: short

distance and high accuracy. Both active sensors can handle lighting changes, their depth measurements are dense (many points in a scanned area), resolution is, according to Zeng et al., relatively low compared to stereo vision and cost is high. ToF sensors suffer from reflection of bright surfaces. The measurement can be distorted when multiple reflections enter the sensor due to concave shapes.

Structured light is a projection of light patterns on a surface [5]. The patterns will deform when touching the curvature of an object. The shape of the object is determined based on this bending or distortion. A stereoscopic sensor is needed to sense the 3D shape. Also the thickness of the line can be used for the distance measurement. The performance of this method also highly degrades in the case of highly reflective surfaces.

Figure 3.1 shows a reconstruction setup with a range sensor and its field of view. Also the different types of areas can be observed, being the scanned surface, empty area, occluded area and occlusion plane. The obtained resolution varies with the surface angle to the sensor. Low angles result in better results.

In the case of scanning aircraft components for maintenance purposes, the distance is relatively short and high to very high resolution is desired [6]. The components can be very reflective and are texture-less. Therefore, stereo vision can be a candidate for application in the industry. However, it is strongly affected by light and depends on texture on the surface of the object. These problems can be solved by using a structured light scanner combined with a projector. To resolve the absence of texture of the surface, a projector with the ability of projecting texture can be used. The LES, explained in Section 3.4, has a similar texture projector.

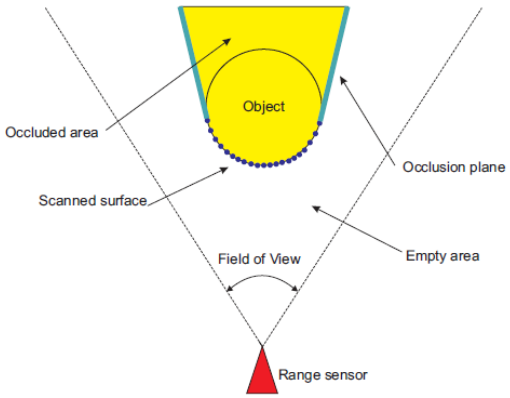


Figure 3.1: *Reconstruction scene [8]*

3.3. Robots

As the aim is automation of processes, robots come to the table, sometimes even literally. Using robots for scanning and reconstructing objects for computer models needs active decisions taken manually via a controller or by the system itself. As more and more automation is preferred and required, the definition active vision comes into play. Robots needs to move autonomously in order to achieve the automation goal.

From literature, two types of robots come to the foreground for 3D reconstruction of objects. These are robotic arms and mobile robots. Robotic arms have a fixed base and an arm to move the sensors to particular poses. In the case of the LES, which includes a robot arm, the base is attached to a linear module which can translate in a vertical direction (or horizontal direction if the setup is rotated). These kind of robots are used for tasks where relatively small movements are required. The sensors are attached to the last part of the arm, called the end-effector. The position of the end-effector is also the position of the sensors. Robot arms usually have several DOFs and rotation capabilities. Being partially static limits the working area of robotic arms, also that of the LES.

In the case when more flexibility is needed and the workspace is large, mobile robots are used. These robots also move sensors, which are placed in their robotic hand or are mounted on top of the head. This type of robots also need path planning to avert collisions with the object or their surroundings. They are prone to accumulating position errors leading to more uncertainty.

In case of very large objects, vehicles or buildings, the usage of drones is also an option. However, deploying flying vehicles can lead to high positioning errors which can accumulate. This will require an additional positioning system.

3.4. LES

The LES consist of a robotic arm (Robolink RL-DC) mounted on a linear sliding module. Two sensors are mounted on the robotic arm which are a stereoscopic 3D depth sensor and a 2D Red Green Blue (RGB) sensor. Parallel to the measurement setup, a wing component of a civil aircraft (Boeing 737) is attached as test specimen. This component is the leading edge slat which is used for increasing the aerodynamic performance of the aircraft.

The current vertical experimental setup has five DOF, four from the robot itself, namely in the x-, y-, z-direction and rotation and one from the linear sliding module, again in the y-direction. Figure 3.2 shows the setup with the robot arm. The user can manually give location inputs to move the robot to a location on the wing.

The monochrome 3D depth sensor is an Ensenso N35 and consists of a left and right camera, both 1.3 Megapixel (MP), for stereoscopic vision to calculate the depth of an object. This provides the distance. The structured light scanner produces a height map with location values. The resolution is 1280 x 1024 pixels. The maximum working distance is 3 m. The optimal working distance of the sensor (ρ) is around 0.4 m and 0.8 m, obtained by tests of objects with different sizes. When the optimal ρ is set to 0.6 m for an object with a volume of 0.6 m width (x-axis) by 0.4 m height (z-axis), the sensed depth in y-direction is 0.3 m (factory specifications). It is expected that defects around 1 mm should will be detected with this parameters. Between the stereoscopic lenses, a blue light projector ensures the illumination of structured light on the surface. The 3D depth sensor also uses its own licensed texture projection to be able to predict the depth from high-reflective surfaces like an aircraft wing. The output of this camera is a point cloud for the 3D model. A second RGB sensor of 5 MP is mounted above the 3D sensor to colorize the point clouds.

Both cameras (Figure 3.3) result in a 3D model after several images which can be analysed by the available industrial computer. The image acquisition is done by software of the camera combined with a custom code written by a former NLR researcher.

Figure 3.2: *The LES set-up*

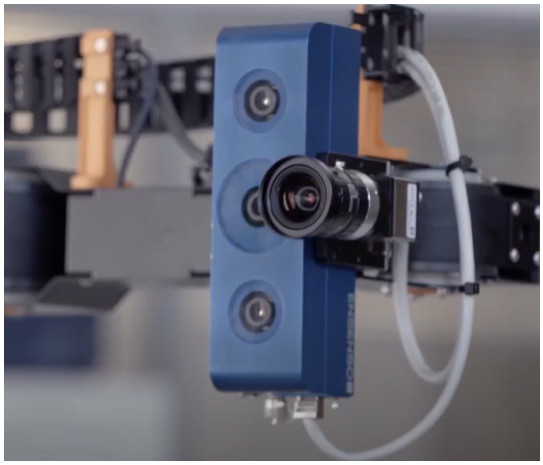


Figure 3.3: *The sensors of LES*

The robotic arm has, when fully unfolded, a length of 0.71 m. This is higher than the distance between the base of the robotic arm and the current test object in the frame (0.47 m), meaning it can not fully unfold at an initial scan. This constraint also restricts the full azimuth angle (ϕ) of +90°(full right) and -90°(full left) on the xy-plane to not be achievable when maintaining a ρ of 0.6 m. Therefore, the angles are limited to +45°(right) and -45°(left) leading to the fact that only the the front part of the object can be scanned, hence the name LES. The current test object also limits the ϕ at +45 °and -45 °to ensure the working distance ρ. Also, only the front surface of objects are considered for research as the back side of the test object is not reachable.

The NBV algorithm is tested in a computerized model of the LES set-up. The main reason is that the experimental set-up has no collision avoidance at this stage. Without this, the LES is prone to accidents.

3.5. Summary of Chapter 3

Reconstructing an object in 3D requires images from multiple viewpoints. These are then merged to build a complete 3D model. This Chapter described three types of sensor types for scanning the surface of an object. These methods are called stereo vision, ToF and structured light. In order to automatically position the sensor to the desired location, a mobile robot or robotic arm can be used. The experimental setup of NLR consists of a five DOF robotic arm with two sensors on board: a monochrome stereo vision 3D sensor, which is a structured light scanner with a projector, and a RGB sensor. The working distance of the LES is 0.6 m, which is the distance where the scan quality is sufficient. The sensor can make azimuth and elevation angles of plus and minus 90°, however due to the fixed test setup and relative short arms, these angles are limited to plus and minus 45°, in order to satisfy the working distance of 0.6 m.

4
Concept of NBV and Algorithms

This Chapter describes the concept of NBV by first introducing the term in Section 4.1. The following Section (4.2) is about the classification of NBV algorithms. Section 4.3 dives into the constraints and requirements of typical NBV algorithms. The next Section (4.4) explains the methods of data acquisition, where the concepts voxel space (4.4.1), triangle meshes (4.4.2) and point clouds come into play (4.4.3). Section 4.5 contains an overview of the applicable algorithms for LES to predict the NBV. Section 4.6 is an evaluation of the applicable NBV algorithms. Section 4.7 summarizes this Chapter.

4.1. Introduction to NBV

Reconstructing an object in a computer environment requires several images taken from different locations, also called viewpoints, to obtain a complete model [7]. The objective is to determine where to look next, meaning where to position the sensor system, after an first initial image of an object is taken. To date, finding an optimal set of viewpoints for 3D object reconstruction by sensors on robots still remains a challenging topic due limitations like occlusion and positioning as well as sensor performance. According to Zeng et al. [1], there were until 2008 more than two hundred research papers predominantly focusing on sensor positioning and view planning.

View planning for 3D object reconstruction has been a problem which has been studied since the 1980s. The common applications of view planning are for object reconstruction, scene modeling, object recognition and pose estimation. This thesis focuses on object reconstruction.

As the to-be-scanned object is unknown, planning the viewpoint beforehand is not possible. Researchers around the world try to find the best solution and are aiming at the most optimum automation of sensor positioning and reasoning where to look next after the first initial scan. This lead to the term NBV, which is widely used in literature. NBV depends on an algorithm, which produces a new best viewpoint in the working space of a sensor system. Chen et al. in their book [8] state that a system without view planning may need as many as seventy different images to completely reconstruct a standard 3D object. Optimal view planning strategy can reduce this number to less than ten.

The reconstruction process consists of the following workflow: positioning, scanning, registration and planning of the NBV [9]. Figure 4.1 summarizes this process. The sensor produces a range image and, from the initial position, a partial model.

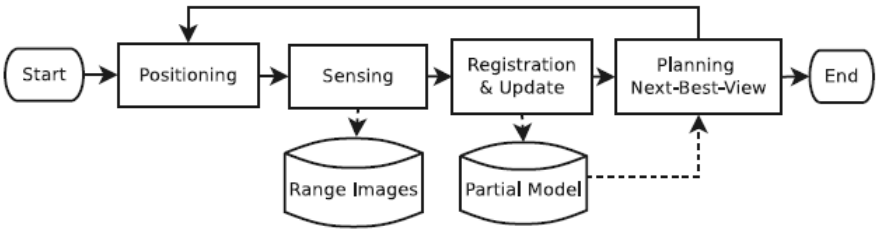


Figure 4.1: *Reconstruction workflow [8]*

The four key challenges in view planning are uncertainty about the working environment, variability of constraints and requirements, positioning error and underperformance of sensing devices [7]. Occlusion hardens data collection as some surfaces are unseen due to shadows, test setup or other elements that block the view. Also the surface material, abrupt changes in shape and texture affect the data acquisition process. NBV requires knowledge about three spaces: the surface space, viewpoint environment and work space [2].

The performance of a view planning strategy and algorithm is mainly measured in the amount of required scans compared to manual operation, surface coverage, quality, digitization time and total travelled distance by the robot. The NBV algorithm should stick to several constraints, which are explained in Section 3.3, and must ensure that all surfaces scanned are reconstructed, even of relatively complex objects.

4.2. Classification of NBV algorithms

NBV algorithms are divided into two categories: model-based and non-model-based. Knowing the dimensions and the object itself beforehand falls under model-based. From [10], it was found that the least number of viewpoints in case of prior knowledge about objects is a nondeterministic polynomial problem, meaning that there is no global optimal solution for most reconstruction tasks. The view planning algorithm should generate a near optimal solution within tolerable time. The non-model-based method does not have information about the object, only a rough estimation of the dimensions.

The two classifications search-based and volumetric, or synthetic, also further divide approaches for finding the optimal NBV algorithm [11]. Search-based approaches make use of large numbers of candidate viewpoints which are then selected under defined constraints. Algorithms are distinguished based on these selections. The goal of search-based approaches is Information Gain (IG) ([12], [13]) and thus obtaining the most amount of new information at the next view. Also the quality of the output ([14], [3]) and cost of robot movement [15] can be taken into account.

Non-model based methods mainly use volumetric or surface reproductions [16]. Volumetric approaches in advance discretize the to-be-scanned field to voxels, being small cubes, and do not depend on the object geometry. The surface method on the other hand analyses the local geometry during the scanning process and is dependent on the geometry.

4.3. Constraints and requirements

It is crucial to know on what bases the algorithms are compared and what the main strategy is in view planning. Early NBV algorithms do not take occlusions into account and also do not have termination criteria based on quality. Later, the quality and running time were also considered.

Many papers have discussed constraints and requirements for realistic NBV strategies. Scott et al. [2] categorized them as general, object, sensor and positioning constraints, summarized in Table 4.1.

From general constraints the requirement of a model quality specification arose. With that the question can be answered as to whether the view planning algorithm should focus on maximum possible quality, precision and complete reconstruction instead of runtime. For other purposes like inspection of particular surfaces, reconstruction of the complete object may not always be needed.

The view planning strategy or algorithm should also be applicable to different sensors, objects and positioning systems. Also sensor parameters like laser intensity or scan distance should be planned through generalized viewpoints. So, it is desired that the NBV algorithm is generalizable for any other similar sensor.

As 3D image acquisition requires multiple viewpoints, obtained images should be merged together. This can only be realized by view or data overlap. The NBV algorithm needs to foresee the amount of image overlap along the edges of adjacent images to be able to integrate the obtained scans from different poses.

The NBV algorithm also needs to be robust to be able to handle system failures, needs to be efficient to compete with the case of manual operation and should stop, self-terminate itself, when the mission is accomplished or even failed in the meantime.

Object constraints are the limited knowledge beforehand about object shapes and material. The view planning algorithm does not have a model database to compare objects and must function for all reasonable geometry as well as materials.

As for the sensor constraints, the performance is limited by the frustum, shadow effect and reflections. [2]

Table 4.1: *View planning constraints [2]*

Category	Requirement/constraint
General	Model quality
General	Generalizable algorithm
General	Generalized viewpoints
General	View overlap
General	Robust
General	Efficient
General	Self-terminating
Object	Minimal a priori knowledge
Object	Shape constraints
Object	Material constraints
Sensor	Frustum
Sensor	Shadow effect
Sensor	Measurement performance
Positioning system	DOF
Positioning system	Pose constraints
Positioning system	Positioning performance

The position system is restricted by the DOF and range. These constraints are evaluated for the feasibility of a viewpoint, meaning can the sensor reach the desired location. Also here performance comes into play: repositioning should happen with minimal pose error and within relatively little time.

The authors of [8] dive further into sensor constraints and describes ten limitations of these sensing devices, being the visibility, resolution, viewing distance (depth), field of view, overlap, viewing angle, occlusion, reachability of the viewpoint, collision and runtime.

According to Pito [17], using a full model of the sensor and its sampling pattern contributes to the effectiveness of the NBV algorithm. This also results in an universal algorithm for other sensors.

4.4. Data acquisition

View planning is based on detection of an object, especially a surface, in a predefined bounding box in 3D space. The NBV is determined during operation of the system and not beforehand. Obtained data from the initial scan is used for determination of the next viewpoint. The view planning strategy is therefore dependent on the method of data acquisition. The leading methods are voxel representation, triangle meshes and point clouds, illustrated in Figure 4.2 [1]. The first two are also called structured representations, the latter unstructured [18]. Mesh triangles are for surface representations and voxel space is for volumetric structures. Ray casting is used for structured methods whereby occlusion and scene coverage can be detected.

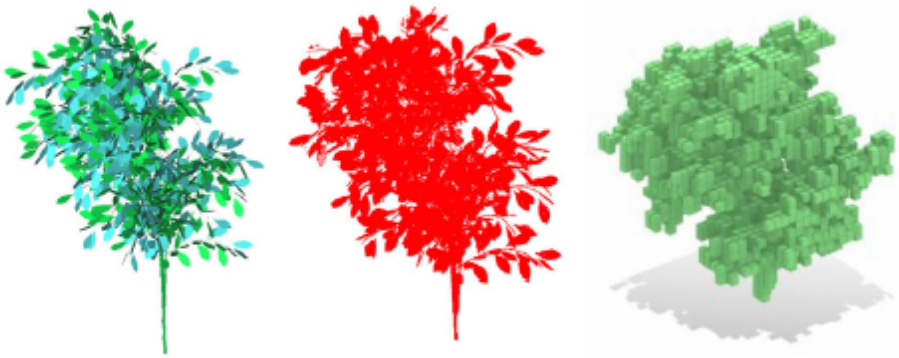


Figure 4.2: *Triangle mesh, point cloud and voxel representations [1]*

4.4.1. Voxel space

One of the strategies to determine the NBV is the division and discretization of the 3D environment by a occupancy map (grid) or an octree [1]. In case of uniform division, the grid is called a voxel map, which is a 3D matrix of voxels [19]. When hierarchical division is executed, the grid becomes an octree [20]. The occupancy map is used to store data about the reconstructed 3D object [21]. The author of the latter reference uses an octomap, which is an probabilistic octree, to deal with imperfect sensor readings. The majority of the view planning methods use the generate-and-test method. This implies that the viewpoint space is discretized and selection of the NBV is done by an optimization algorithm. Representation by voxels is seen a simpler than other representations.

Connolly [22] was one of the first to find a way to digitize an object in a predefined space for reconstruction. He introduced the voxel space and came up with the idea of an octree structure, basically a tree which recursively divides a cube into eight smaller cubes. This structure contains four nodes, divided in a parent and their children. The leaves of this tree are categorized as empty, occupied and unseen. Empty is describing empty space which is visible from at least one of the viewpoints used to create the octree. Occupied nodes are detected points on a surface of an object and unseen nodes are not (yet) sensed. During initialization of the process the root of the (oc)tree is set to unseen. View selection is based on the coverage of the most unknown voxels.

The major disadvantage of voxel representation is memory consumption as it processes the complete working environment leading to increased computation costs and time. A standard object only occupies a relative small part of an environment, a surface even smaller. As multiple scans are required to reconstruct an 3D model and voxelization or an update after every scan, the computation load and time increases even more.

4.4.2. Triangle mesh

One of the common techniques to reconstruct the surface of an object is by triangle meshes. These triangles are attached to each other by their edges or corners. Triangle meshes provide fine details of surfaces. A shortcoming of this method is that the empty or unknown space is not processed, which voxel representation does.

4.4.3. Point cloud

Most range sensors use the point density, or cloud, method to collect data and obtain surface details. Consider a point p_i in 3D, given by the coordinates x_i, y_i and z_i. These points are spatial coordinates and the values could represent the surface reflectivity and, when an RGB sensor is present, the colour. Although the point cloud method is similar to the triangle mesh and also cannot process empty and unknown spaces, the data of point cloud is seen more easy to use. The point cloud method can restore more surface details compared to the voxel strategy. However, occlusion detection is not possible with point clouds without additional strategies as it excludes the usage of ray casting.

The point cloud representation does not require a pre-discretization of the scene. Also this unstructured method is linked to the number of scans instead of the scene size, which voxels are. Therefore, this data method also can be used for large scale objects.

4.5. Overview of NBV algorithms

In order to gain a knowledge of NBV methods and their properties, fifteen algorithms are theoretically explained in chronological order.

Some of the earliest algorithms are that of Connolly in 1985 [22], where he describes his determination of NBVs. The researcher explains two algorithms, the planetarium algorithm and the normal algorithm, which is being referred to and seen as a first basis in many papers afterwards. The planetarium algorithm, which sets up an evenly sampled sphere around the object (Figure 4.3), provides the NBV based on testing of a set of views around the object. The number of the so called empty nodes, seen from a viewpoint, is determined by ray casting. The direction with the largest amount of empty nodes is chosen as the NBV. The center of the object is matched with the center of the sphere. The normal algorithm uses the sum of the areas of all six faces

of the cube and determines the NBV via the normals of unknown voxels, which can be seen in Figure 4.4. In more detail, the best next location is found by determining the cube vertex whose three adjoining faces, with the tag unseen or empty, have the highest values. Figure 4.5 contains a part of his normal algorithm. This algorithm has only eight possible NBV positions.

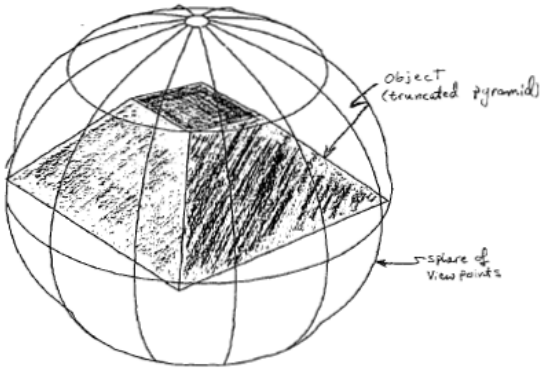


Figure 4.3: *Spherical representation of Connolly [21]*

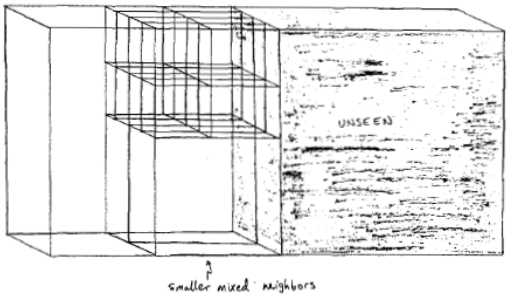


Figure 4.4: *Unseen nodes and empty neighbours [21]*

For each node N of the octree:
If N is an *Unseen* node, then
For i from 1 to 6:
If $neighbor_i$ is *Empty*, then $count_i = count_i + 4^{-j}$ where j is the level of node N.
else if $neighbor_i$ is a *Parent* node, then
for each descendant, M, of $neighbor_i$ which abuts the original node N:
if M is *Empty*, then $count_i = count_i + 4^{-k}$, where k is the level of node M.

Figure 4.5: *Description of the normal algorithm [21]*

Maver and Bajcsy [23] used a range scanner and a rotating table in 1991 as the basis for the object to solve the NBV problem. As the difficulty of finding the NBV lies in the unseen sides of the to be modelled object, their suggestion was to model them as two-and-a-half-dimensional (2.5D) polygons. They gave each edge of a polygon a height value which corresponds to the median of the height of the used pixels. The proposed method is to scan these polygons from unoccluded viewpoints. As the table used as a basis could rotate, angles with clear sights into unknown parts in the view space were summed up in a histogram. The decision of the NBV is related to the largest angle.

In 1995, Yuan proposed a NBV algorithm [24] using a set of represented surfaces to estimate the visible direction of unseen surfaces. The obtained image from the sensor was segmented to form patches of the surface. The author used the proven knowledge that the total Gaussian mass of surfaces of a convex object must be zero. Also the usage of a total mass vector chain is emphasized, which is a sum of all mass vectors. Mass vectors which form a closed chain, that is when the mass vector sum is a zero vector, indicate that a surface boundary is closed. The NBV is calculated in opposite direction of the mass vector. This is visualized by Scott in his paper [2] in Figure 4.6. The Gaussian sphere can also be used for concave surfaces. However, concave surfaces most of the time do not form a closed Mass Vector Chain (MVC). A virtual surface patch is needed to represent a cutting plane.

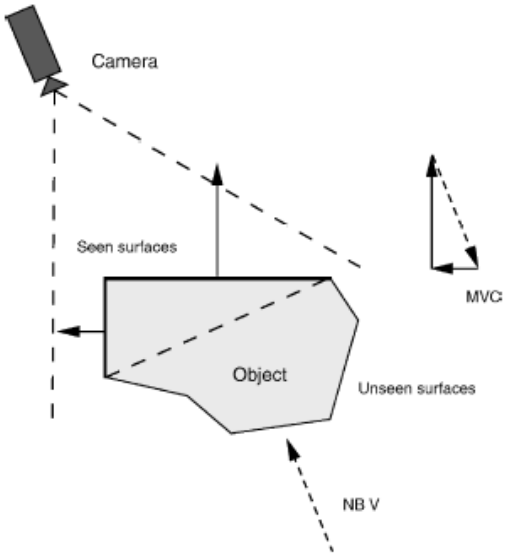


Figure 4.6: *Visualization of the mass vector chain (MVC) [2]*

A quite unique setup for his time was of Papadopoulos-Orfanos and Schmitt in 1997 [25]. They used a turntable for the object and three linear sliders for the sensor. The authors explained the relationship between the intrinsic properties of an object, obtained by mathematical morphological operations, to its digitization by using shapes or slices. The proposed algorithm is based on voxel representation. The workspace containing the object is labelled as unknown. The used laser rangefinder as the triangulation sensor has a small field of view and a working distance between 5 cm and 10 cm. Due to these relative small distances between the object and the sensor, the authors also taught about collision avoidance. The hierarchical algorithm takes into account occlusion and is divided in two parts: path planning and sensor planning.

The algorithm of Papadopoulos-Orfanos and Schmitt starts, at low level, with the digitization of a single view. A so called z-buffer is used for representation of the scene. Calculations for path planning are in 2.5D. The unknown space is explored layer after layer in a zigzag pattern, described in Figure 4.7. In case of a surface voxel, the system avoids collision by changing the movement direction. The second stage of the algorithm is at high stage for choosing the NBV. The object on the turntable is rotated around the vertical axes in steps of $2/n$, where n is a chosen parameter for digitizing views (n), for digitizing n views.

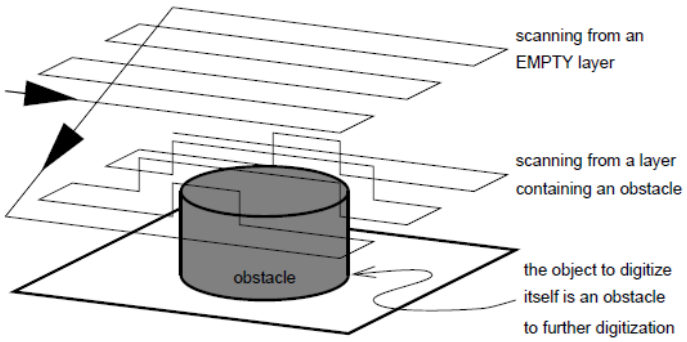


Figure 4.7: *Workspace exploration by layers [25]*

Pito [17] came up with an algorithm which is often referred to in literature. He in 1999 further advanced the work of Connolly by considering several constraints, requirements and self-termination. His algorithm based om triangular meshes did take into account occlusions and output quality. Merging a set of range images was considered by Pito by using an overlap constraint. He emphasized that algorithms which do not address an overlap constraint can produce unreliable results unusable for further reconstruction and

processing. Pito calls his algorithm the PS Algorithm and divides the viewing volume in seen and unseen areas by using Ranging Ray (RR). Areas that are not sampled are tagged as the void volume. The term void surface is used for the separation of the void volume from the rest (Figure 4.8).

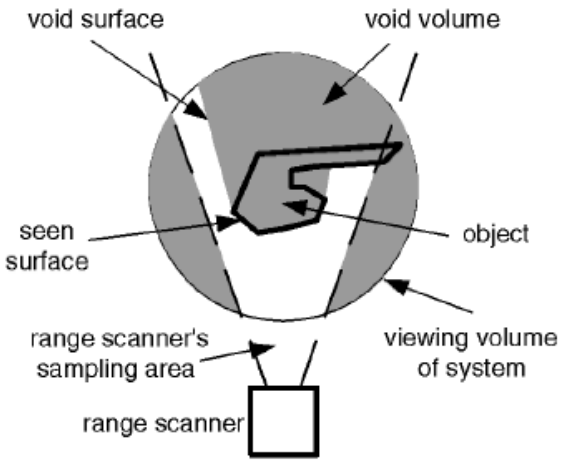


Figure 4.8: *Viewing volume described by Pito [17]*

Pito determines the next position by patches, which run from the edges of an object. An analysis of the neighbourhood decides which patch belongs to a hole in the model or is an edge of the object. Each edge has a local coordinate system. The determination of the NBV relies on maximizing an objective function. A workspace of 30 cm by 30 cm is mentioned.

The method of Reed and Allen [26] is nearly similar to the algorithm of Maver and Bajcsy and has a more general solution. The surfaces in the viewing volumes are divided in empty areas and unknown regions. These surfaces, also called targets by the author, were determined by volumetric representations of each image. Each surface triangle was transformed into a prism. Solids were created thanks to the obtained 2.5D data. The solids cross each other and form groups of triangles (mesh), tagged measured or occluded. Occluded surfaces are used for calculating the next view. The view vector is placed perpendicular to these surfaces.

In the same year, Banta et al. [27] tried to solve the NBV problem by using a voxel space representation. He based his work on the planetarium algorithm of Connolly [22], which means that he also placed a sphere around an object on a self-determined distance. Each voxel of same size is named surface or occluded. Three methods are proposed to determine the NBV. The first effort is by detecting the edges of the object, second being finding unknown areas in the working space and third by combining voxels which are unknown. The iteration of the observation angle is linked to occlusion detection. A self-termination criterion is present.

In 2004, Callieri et al. [28] presented a NBV algorithm consisting of two phases. Firstly, an initial and unfinished model of the object is obtained via a fixed set of view positions. The second step is detecting discontinuities on the rough model, which are then filled up. This method was then seen as novel as the 3D digitization used different colours for the surfaces and background. The discontinuities are identified counting pixels with deviant colours.

Chen and Li [29] also analysed a partially obtained model. The best next viewpoint is acquired by searching for the object side with the least contour change by calculating the surface trend. The surface order correlates to the smoothness of a surface. The exploration direction is obtained from the area with the lowest surface order for gaining most information about unknown areas. According to Chen and Li, it is only required to compute the surface order near the boundary of the known region.

The predictive trend curve for a tea cup on its side is depicted in Figure 4.9. The algorithm searches for a viewpoint where the area of the surface can be maximized. This can be done accurately when the surface of the model has first and second order curves and surfaces. The modeling process of Chen and Li is shown in Figure 4.10 in and consists of the acquisition of the view (S1), reconstruction of the local 3D model (S2),

registration and merging with the global model (S3), an analysis part of the model and checking of termination criteria (S4), ranking of the directions (S5), trend surface computation and NBV determination (S6) and robot movement (S7). Self-termination is achieved based on a completion criterion after estimating the sizes of uncertain areas via another algorithm.

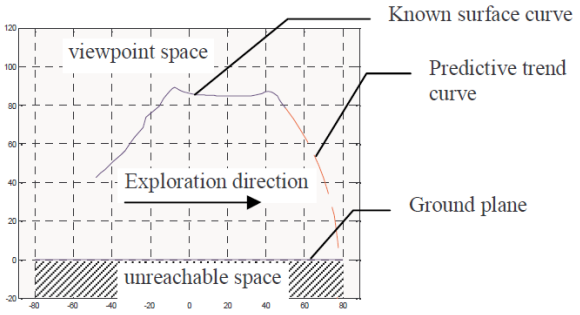


Figure 4.9: *Predictive trend curve [29]*

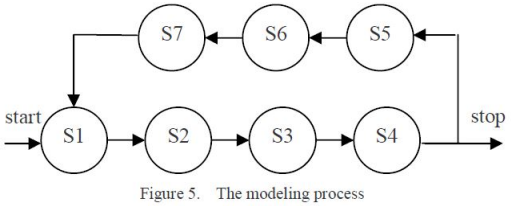


Figure 5. The modeling process

Figure 4.10: *The modeling process of Chen [29]*

The algorithm of Chen and Li has issues at surface edges and boundaries, where the curvature change abruptly resulting in computation errors.

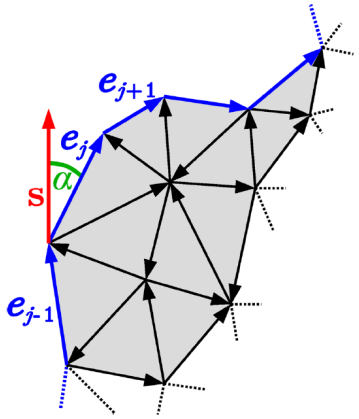


Figure 4.11: *Detection of a left boundary in the mesh [30]*

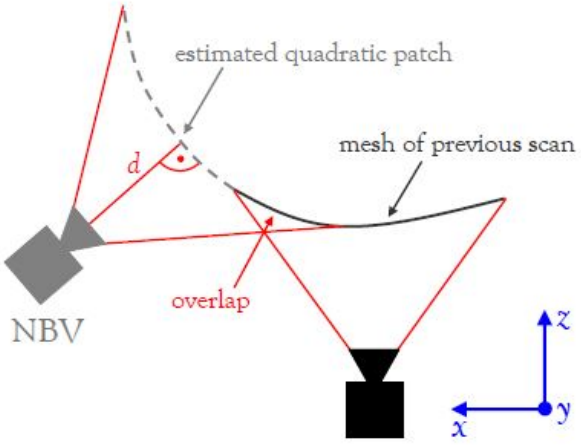


Figure 4.12: *Predicted NBV based on quadratic patch [30]*

In 2011, Kriegel et al. [30] based their algorithm on the detection of unmatched edges in a partially digitized model. The author used triangle meshes and classified the edges of these triangles as normal or border edges. Groups of border edges are classified further as left, right, bottom and top. Figure 4.11 shows the detection of a left boundary in the mesh. The observation angle (α) is between the scan direction (s) and an edge and its component (e_i). It is calculated by Equation 4.1:

$$\alpha = arccos(\frac{se_i}{|s||e_i|}) \tag{4.1}$$

The surface trend is calculated quadratically for the point with coordinates x, y and z (p_i, x_i, y_i, z_i):

$$p_i = [x_{p_i}, y_{p_i}, z_{p_i}]$$

by the equation, including the constants of the quadratic function (a, b, c, d, e, f):

$$z_{p_i} = f(x_{p_i}, y_{p_i}) \;=\; ax_{pi}^2 + bx_{p_i}y_{p_i} + cy_{pi}^2 + dx_{p_i} + ey_{p_i} + f \tag{4.2}$$

Equation 4.2 is of low order which reveals if the curvature at the boundary is curved inward of outward towards the unknown region.

The normal vector (Equation 4.3) arises from the derivative of the surface trend and is used for the observation angle.

$$n_i = \begin{pmatrix} x_{n_i} \\ y_{n_i} \\ z_{n_i} \end{pmatrix} = \begin{pmatrix} \frac{\delta f}{\delta x_{p_i}} \\ \frac{\delta f}{\delta y_{p_i}} \\ -1 \end{pmatrix} = \begin{pmatrix} 2ax_{p_i} + by_{p_i} + d \\ bx_{p_i} + 2cy_{p_i} + e \\ -1 \end{pmatrix} \tag{4.3}$$

The z-component is set to -1 as it is pointing towards the sensor, which is opposite to this vector component. The predicted NBV is depicted in Figure 4.12, d being the sensor distance.

The algorithm of Kriegel et al. contains an overlap constraint where the size of the overlaps is changeable. Self-termination is available when there are no border edges left in the mesh. Kriegel claims that his universal algorithm can be used with any range sensor. Two years later, in 2013, Kriegel selected his NBV based on the two 3D models probabilistic voxel space and triangle mesh, so by combining the surface and volumetric representation. The modeling process is shown in Figure 4.13.

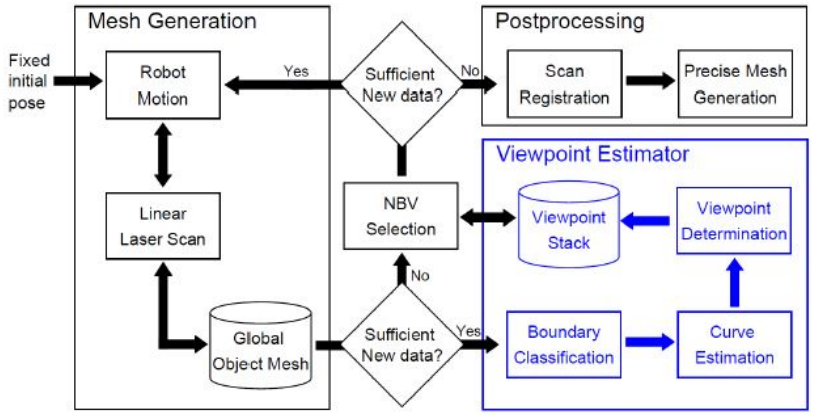


Figure 4.13: *The modeling process of Kriegel et al. [30]*

Another algorithm based on border detection is that of Karaszewski et al. in 2012 [31]. He identifies those regions by best fitting a plane. The centre of this plane along its normal vector is the direction where the sensor is positioned to.

A NBV algorithm which identifies the borders of a partially scanned object is that of Khalfaoui et al. [32]. He in 2013 separated each directional measurement and analysed the angle between the normal of the scanned surface and the vector in the observation direction. Large areas are tagged as barely visible and are cumulated. The sensor is placed perpendicularly to these areas. There is no occlusion handling or self-termination.

Ahmadabadian et al. in 2014 [33] presented a two stage view planning strategy based on triangle meshes. The first one is for acquiring the first mesh from the initial view. In the second stage, positions and orientations of a head model are calculated. The object is enclosed by an (calculated) ellipsoid instead of a sphere, as can be seen in Figure 4.14. The NBV is selected as a set of similarly oriented views for getting the observation angle parallel to the normal vector of the surface.

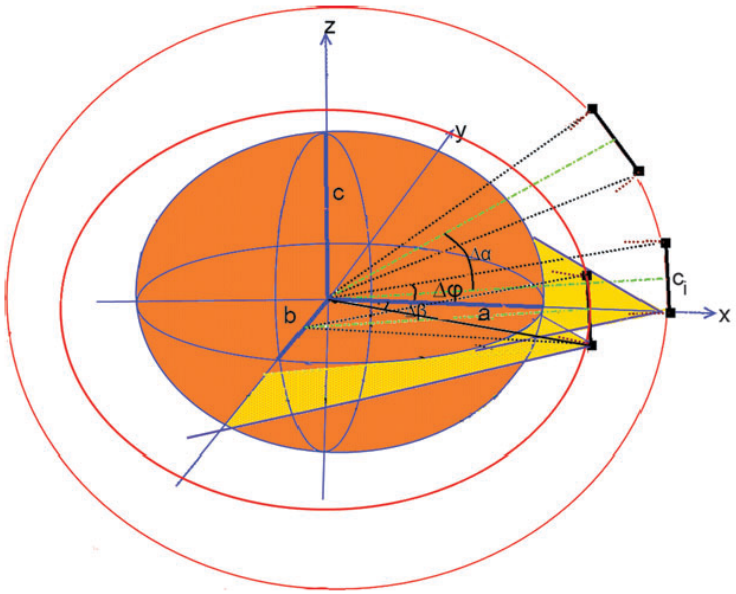
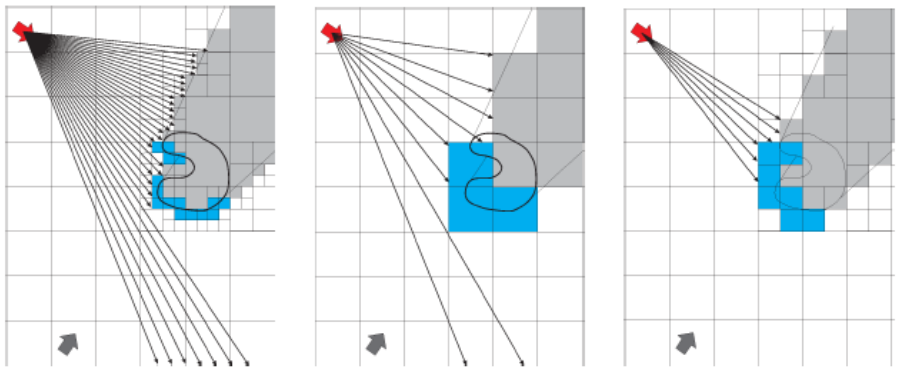


Figure 4.14: *Elliptical enclosure of an object [33]*



(a) Uniform ray tracing. All sensor rays are traced in the octree.

(b) Rays traced in a coarse octree.

(c) Ray tracing in a finer octree only for touched voxels.

Figure 4.15: *Ray tracing methods [8]*

In the same year, Vasquez-Gomez et al. [9] based his volumetric and search-based algorithm on an improved version of the planetarium algorithm of Connolly. He deviated from the sphere into a shape more similar to an icosahedron, a polyhedron with twenty faces. Vasquez-Gomez used voxels instead of triangle meshes or point clouds. He divided the voxels into five: unknown, used, empty, occluded and a 'occplane', an occluded plane. Each voxel features a normal vector and has a quality characteristic. This quality condition is calculated by taking the maximum cosine of the observation points inside the voxel.

Vasquez-Gomez et al. added knowledge of the so called ray tracing algorithm of Bresenham [34] in his method. Overlaps, the varying distance of the sensor to the object and the observation angle are taken into account. In comparison to Kriegel et al. [30], who proposed the use of IG, Vasquez-Gomez et al. used measured unknown surfaces to evaluate the views. The views are selected with five characteristics, being new information, a positioning, sensing and registration constraint and cost of the robot movement. Candidate views are evaluated by a visibility computation over the octree, using Uniform Ray Tracing (URT), after which they are ranked by an utility function. The use of a probabilistic octree is stressed in [35] to cope with sensor noise. Hierarchical Ray Tracing (HRT), where resolution is locally increased when rays touch voxels, is used

to speed up processing. Figure 4.15 clarifies the comparison between URT and HRT. Views are generated directly and each iteration is a set of random and uniformly generated samples. Latter are filtered to obtain useful views. The researcher generated 10.000 random samples per iteration and used only a few hundred for further processing. Rays from the sensor which do not intersect the sphere around the object, are discarded.

The NBV algorithm of one of the works of Vasquez-Gomez et al. [21] generates set of candidate views (V). The viewpoint (v) is denoted as V. Each v is a data structure containing the robot configuration (q), the workspace of the sensor (x), a homogeneous transformation matrix (H) and the utility function (u) of the view. This tuple can be written as $v_i = (q_i, x_i, H_i, u)$. The goal is to select $v \in V$. The candidate views are ranked using u, given by equation 4.4:

$$u(v) = pos(v) \cdot reg(v) \cdot sur(v) \cdot dist(v) \tag{4.4}$$

The positioning constraint (pos) has the value 1 when all voxels in the Robot Bounding Box (RBB) are free. If not, this value is 0. The registration constraint (reg) is 1 when a minimum overlap is present, otherwise 0. surface information (sur) is for the new surface information and depends on how much unknown voxels from a view are present. The unknown voxel can also represent an occluded surface. sur is obtained from equation 4.5 using the amount of unknown voxels (un_0) inside the OBB:

$$sur(v) = un_0 \tag{4.5}$$

The position constraint ($dist$) are candidate views which are evaluated based on their distance to the current view. Equation 4.6 describes $dist$, where the node distance (ρ_{nod}) is used between nodes of the current robot position ($v.q$) and candidate position candidate position ($p.q$).

$$dist(v) = \frac{1}{1 + rho_{nod}(v.q \cdot p.q)} \tag{4.6}$$

Algorithm 1: NBV for 3D Object Reconstruction

Input : Initial position (p_0)
Output: Object model (M)

Configure octree and rays-tree;
$p \leftarrow p_0$;
while *true* **do**
  Plan a path for reaching p;
  Move the robot to p;
  $z \leftarrow$ Take range image;
  $M \leftarrow$ Update model M using z;
  $V \leftarrow$ Generate candidate views;
  foreach $v \in V$ **do**
    if $pos(v)$ **then**
      `HierarchicalRayTracing`(M,v);
      if $reg(v)$ **then**
        $v.u \leftarrow u(v)$
      else
        delete v;
      end
    else
      delete v;
    end
  end
  $p \leftarrow$ Best evaluated view from V ;
  if p *does not provide information* **then**
    Return M;
    Finish reconstruction;
  end
end

Figure 4.16: *NBV algorithm of Vasquez-Gomez et al. [21]*

Algorithm 2: Hierarchical Ray Tracing (HRT)

Data: v, M, k, $rayNode = root$
Result: $VoxelAmouts$

foreach $r \in rayNode.C$ **do**
  switch `CastRayIn`$(r.u, M, k)$ **do**
    case *OCCUPIED*
      if $k == d_{max} \vee$ `isLeaf`$(rayNode)$ **then**
        $VoxelAmounts.occupied$ ++;
      else
        Add `HRT`(v, M, $k+1$, r) to $VoxelAmounts$;
    case *UNKNOWN*
      $VoxelAmounts.unmark$ += $r.lf$;
Return $VoxelAmounts$;

Figure 4.17: *HRT used for algorithm of Vasquez-Gomez et al. [21]*

Figure 4.16 contains the algorithm of of Vasquez-Gomez et al. in one of their work [21]. The input is the initial position and the output is the object model M. The robot is moved to a position (p), a range image (z) is taken, the model (M) is updated using z and candidate views V are generated.

Figure 4.17 is the second part of the algorithm. Now, the candidate view, octree, ray-tree and initial depth are the inputs. The algorithm produces 'VoxelAmounts', which describes the amount of each voxel type. Finer ray tracing is applied if a voxel is hit.

In 2016, Karaszweski et al. compared thirteen NBV algorithms with various 3D scanners and manipulators. His work is one of the most well explained and detailed demonstrations. The thirteen algorithms that Karaszewski et al. compared are that of Connolly [22], Yuan [24], Papadopoulos-Orfanos and Schmitt [25], Pito [17], Reed and Allen [26], Banta et al. [27], Callieri et al. [28], Chen and Li [29], Kriegel et al. [30], Karaszewski et al (own work) [31], Khalfaoui et al. [32], Ahmadabadian et al. [33], and Vasquez-Gomez et al. [9].

Karaszewski et al. compared the algorithms based on the four criteria: number of scans, reconstruction time, total travelled distance and surface coverage. Also, three types of sensors are used, one with low resolution Microsoft Kinect range camera (MK), a middle version Middle Resolution Sensor (SL1) and the last being a high resolution sensor High Resolution Sensor (SL2). The MK has a large working volume of 87 cm by 63 cm at a distance of 80 cm. SL1 is a structured light 3D scanner using two 2 MP PointGrey cameras and a portable Toshiba DLP projector with a medium combined work volume of 18 cm by 12 cm by 11 cm at a minimum working distance of 40 cm. SL2 is also a structured light scanner using a Canon 60D DSLR-camera and a Casio projector. SL2 has a small working volume of 5 cm by 5 cm by 12 cm at a minimum working distance of 50 cm.

Karaszewski et al. divided their experiment in three parts where the term OSWV comes into the foreground. The three cases are for 3D models with an OSWV of lower than 1, around 1 and larger than 1. He bases his conclusion on simulations of all the NBV algorithms tested on five test objects with different occlusions, cavities and sharpness of the edges.

Karaszweski et al. used two sensor models, one Simple Model (SM) and one Advanced Model (AM). SM is described by the working volume, working distance and maximum observation angle. For more accurate modeling, AM is used where two sensors, separated by a distance called base, are described by a combined working volume, working distance and two observation angles.

Table 4.2 summarizes all algorithms and their properties, varying from the method used to the type of data representation.

Table 4.2: *Overview of NBV algorithms*

Year	Researcher	Property
1985	Connolly [22]	One of first, planetarium and normal algorithm
1991	Maver and Bajcsy [23]	Rotating tables – 2.5D polygons
1995	Yuan [24]	Mass vector chains
1997	Papadopoulos-Orfanos and Schmitt [25]	Small working volume, two staged and turntable
1999	Pito [17]	Occlusion reckoning, quality and self-termination
2000	Reed and Allen [26]	More general volumetric representations
2000	Banta et al. [27]	Voxel space representation, based on Connolly
2004	Callieri et al. [28]	Discontinuities, counting pixels with deviant colours
2004	Chen and Li[8]	Surface trend analysis, contour change
2011	Kriegel et al. [36]	Unmatched edges, triangle meshes, edge qualification, IG
2012	Karaszewski et al. [37]	Border detection, best fitting plane
2013	Khalfaoui et al. [32]	Border detection, separated process, angle between normal and vector
2014	Ahmadabadian et al. [33]	Two stage, triangle meshes, ellipsoid instead of sphere
2014	Vasquez-Gomez et al. [9]	Volumetric ray tracing, HRT, observation angle, CU vs IG

One of the most recent NBV algorithms is that of Border et al. [16]. It is called Surface Edge Explorer (SEE) and plans the NBVs directly from 3D observations. The algorithm uses the surface approach and represents the data as a unstructured density, meaning it uses points instead of voxels or a mesh. SEE observes the scene by searching for frontier points, which are on the boundary. Core points are inside a defined region and outliers are outside that region. The local surface is estimated from the frontier points and its neighboring points. This is illustrated in Figure 4.18. The number of neighbor points (k), the minimum number of neighbor points (k_{min}) radius (r) and desired point density (R) are used in Equation 4.7:

$$k_{min} = \frac{4}{3} R\pi r^3 \tag{4.7}$$

The frontier points result in the orthogonal boundary vector (e_b), frontier vector (e_f) and normal vector (e_n). The normal component is oriented normal to the frontier point (outside this page) and is also in the sensor direction, the boundary vector follows the frontier line, thus the boundary, and the frontier vector is perpendicular to the frontier line (Figure 4.18). From now on the term boundary point will be used for frontier point.

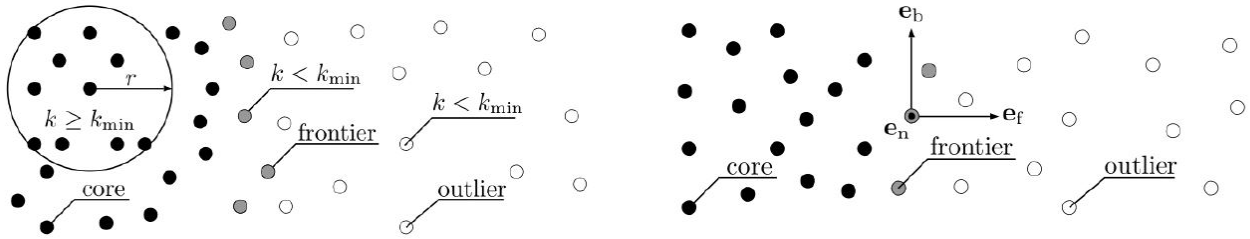


Figure 4.18: *SEE - classification of points and vectors [16]*

4.6. Evaluation of NBV algorithms

Connolly [22] claims that the normal algorithm is faster than the planetarium version. However, the second algorithm has problems dealing with occlusion. The first one is affected less by this problem. Both algorithms do not take into account sensor position constraints, which are present in real experiments. Also there is no information about object size compared to viewpoint distance.

The algorithm of Yuan [24] is applicable to simple objects and is able to make adjustments for occlusion. A major disadvantage of this NBV method is that the sensor positioning system should relocate itself in opposite directions, increasing the travel distance and runtime. Also not all robotic arms are capable of moving to opposite directions.

The solutions produced by the algorithm of Papadopoulos-Orfanos and Schmitt [25] are seen as very simple and triggered comments if they are applicable to objects in real life [37]. Nevertheless, it can be seen as a basis for sensors with a small working volume.

The algorithm of Pito [17] theoretically self-terminates. Projecting a Observation Ray (OR) through the viewing volume takes a long time. The algorithm is generalizable to any range scanner.

The paper of Callieri et al. [28] is unclear about how the view candidates are calculated, leading to the assumption that the new viewpoints are generated randomly.

The method of Chen and Li [29] works for slowly changing surfaces. When strong curvatures or sharp edges are present, this method is too simple. As for the result, the number of automatic scans is larger than a manual operation. There is no information about occlusion handling. However, the researcher of [37] implemented a occlusion detection method from another algorithm.

Kriegel et al. [30] claims that their algorithm is universal and is applicable to any 3D sensor. However, it remains a question if he accounts for occlusion. Their self-terminating Next-Best-Scan (NBS) is considering surface quality. In their experiment, they compares their own older IG method with a new method called IG/Quality. Their conclusion is that his robot was 11 times faster than manual, by hand, scanning. They acquired high-quality models of nine objects between 1 and 10 minutes.

The results of Karaszewski et al. are summarized in Table 4.3 [37], where two plus signs stand for good performance, one plus sign for sufficient, one minus sign for underperforming and two minus signs for very low performance. The performance of thirteen algorithms are compared for different Ratio of object size to scanner working volume (OSWV) values. The evaluation is based on number of scans, runtime, distance travelled and surface coverage.

Table 4.3: *Performance of NBV algorithms based on surface coverage and runtime [37]. Performance is ranked from - - to + + (very poor to very good)*

Algorithm	OSWV«1	OSWV = 1	OSWV»1
Connolly (1985) [22]	+	-	–
Yuan et al. (1995) [24]	++	-	–
Papadopoulos-Orfanos et al. (1997) [25]	++	++	+
Pito (1999) [17]	+	-	–
Reed (2000) [26]	+	+	+
Banta (2000) [27]	++	-	–
Callieri et al.(2004) [28]	++	+	+
Chen et al.(2005) [29]	++	++	++
Kriegel et al.(2011) [30]	+	++	++
Karaszewski et al.(2012) [37]	++	+	+
Khalfaoui et al.(2013) [32]	++	++	+
Ahmadabadian et al. (2014) [33]	++	+	–
Vasquez-Gomez et al. (2014) [9]	++	-	–

Karaszewski et al. underlined that the simple methods of Connolly, Pito and Banta et al. failed in obtaining full 3D models of the self-occluding objects and objects with deep cavities. The mass-vector-chain of Yuan used more sensor movements than the other methods. The algorithm of Papadopoulos-Orfanos and Schmitt yielded good surface coverage. Algorithms which use border detection methods produced successful results, meaning less scans and good coverage compared to simple algorithms. Due to the low resolution of MK, many models had discontinuities. This cannot be attributed to the algorithm and can be solved with sensors with a better resolution, the researcher emphasized.

As for the OSWV around 1 case and sensor MK, all algorithms made use of a complicated set of sensor positions in the sphere. As some parts of the object fell outside the working range of the sensor as the sensor was directed to the center of the object and at no other point, so all algorithms had problems constructing full models. The differences between the 3D models mostly occurred due to the different placement of the objects. Newer models of after 2000 performed better, except that of Banta et al. and Vasquez-Gomez and Schmitt. The latter two are based on the viewing sphere method. All algorithms also had problems with sharp edges calculating faulty viewpoints and resulting in holes in the 3D model. This is partially solved by other scans. The results of Papadopoulos-Orfanos is acceptable at a cost of more required scans, underscored Karaszewski et al.

In the case where the OSWV is larger than 1, some algorithms completely failed. The four algorithms Connolly, Pito, Banta et al and Vasquez-Gomez et al. did not produce acceptable results for any of the five objects. This poor performance is related to the unchangeable distance between the sensor and center of the objects in combination with the shallow working volume of the sensor. The algorithm of Yuan, the mass-vector-chain, also failed due to the wrong selection of the NBV. All algorithms that analyze borders succeeded in producing acceptably results and complete models.

The number of required scans and total travelled distance between the succeeding algorithms for especially complicated objects was quite different. The algorithm of Khalfaoui et al. was in comparison with its opponents little underperforming due to its selection of NBV based on the angle between the surface normal and observation vector. Also at the case of an OSWV of larger than 1 and high resolution sensors, all algorithms were affected by sharp edges. It remains extremely difficult to accurately predict the direction of the surface when there is a sharp edge meaning a major change in direction. In the situations that the algorithm did not self-terminate as it should be, which can be attributed to the used setup, the algorithms were stopped manually.

Karaszewski at al. summarizes their work that there are no NBV algorithms which perform outstanding for all test cases. However, the algorithm of Chen et al. and Kriegel et al. are performing very well in terms of coverage and number of scans for all cases. The third good performing algorithm is that of Karaszewski et al. Although their algorithm required more scans than average in their tests, the coverage and fitting of holes is acceptable. The algorithm of Kriegel et al. and Karaszewski et al. have problems with occlusions, producing invalid new viewpoints. The early and constrained algorithms, that of Connolly and Pito, tend to produces less complete models in the case of complicated and self-occluding objects. The mass-vector-chain method of Yuan does not work very well for OSWV values of 1 and more than 1. The method of Papadopoulos-Orfanos and Schmitt (1997) often misses details, however, it works for all cases. The two-stage method of Callieri et al. and Ahmadabadian et al. resulted in more scans and results of poor quality.

In their own experiment, Vasquez-Gomez et al. used a utility function to account for the quality of the reconstructed model. Using a ray tracing algorithm, they take observation angles and overlaps into account. They also claim to have one of the first methods that determines the NBV directly in the configuration space. Their experiment was compared with other works and the result is that he qualitatively measures unknown surfaces, overlap and movement costs and also efficiently evaluates candidate views. They conclude that, quantitatively, his algorithm achieves same coverage in less time compared to other relevant works. The algorithm self-terminates.

Border et al. [16] emphasized that SEE outperforms most of the state-of-the-art volumetric approaches, among which that of Vasquez-Gomez et al. and Kriegel et al. They simulated seven volumetric approaches and SEE for different objects of the two sizes 1 m and 40 m (4.19). As a result, it became clear that SEE requires less computational time for view planning and result in better surface coverage in less views. SEE also obtain high surface coverage for complex objects, also that of large scales. Their algorithm is a also occlusion-friendly, meaning SEE also can adjust its views in case of occlusion. Border et al. claim from their simulation that the volumetric approaches performed worse in case of multiple self-occlusions as they can not adjust their views to account for occlusion. The computational complexity of volumetric approaches lies in its ray tracing of a voxel grid from every view on a sphere. Also volumetric approaches are limited in scene size and do not size well.

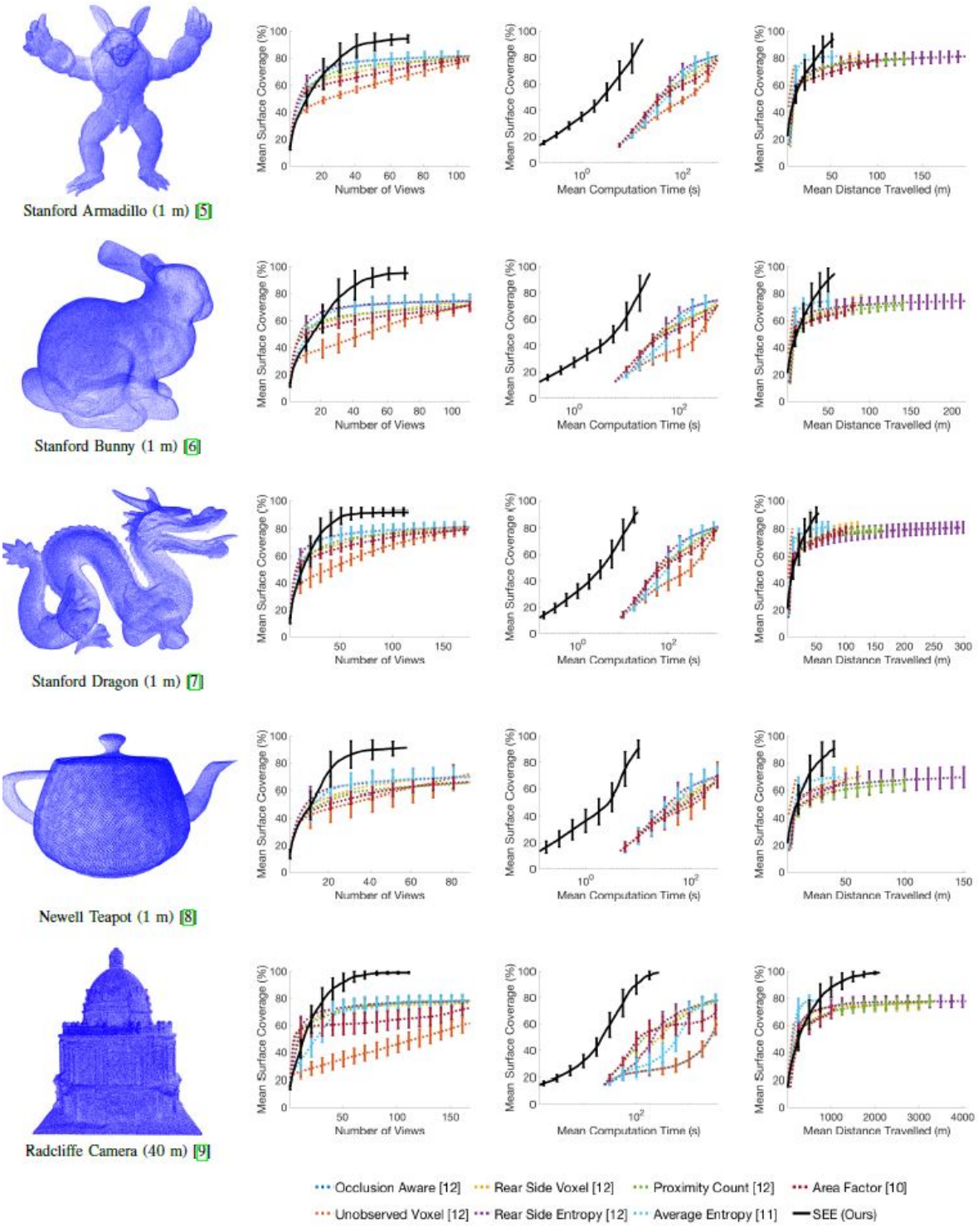


Figure 4.19: *Comparison of SEE with different volumetric methods [16]*

4.7. Research- and sub-questions

The required algorithm for the automation of LES, obtained knowledge about NBV algorithms and results of relevant experiments described in previous Sections lead to the research question:

What is the optimal NBV algorithm for 3D reconstruction of unknown objects with the LES?

This research question is answered with the aid of answers to the following three sub-questions:

(1) How is the imaging pipeline set up for image acquisition and object registration?

(2) What sensor systems are required for unknown object reconstruction?

(3) What are the prominent view planning algorithms?

Also, answers on all above questions support the reasoning behind the choice of a NBV algorithm with the reasoning behind ranking and choosing the candidate points as the NBV. The demonstration of the effectiveness of the optimal NBV algorithm for the LES is performed by a simulation in MATLAB.

4.8. Summary of Chapter 4

NBV algorithms are divided into two categories: model-based and non-model-based. The two classifications search-based and volumetric also further divide approaches for finding the optimal NBV algorithm. Search-based approaches make use of large numbers of candidate viewpoints which are then selected under defined constraints. Volumetric approaches voxelize the working area and perform raycasting. Their memory consumption is relatively high. A NBV algorithm has general, object, sensor and positioning constraints. The commonly used representations for data acquisition are voxel representation, triangle meshes and point clouds. Several prominent NBV algorithms are theoretically compared in order to gain knowledge about the available methods. Connolly was one of the first describing his method of predicting the NBV by a spherical enclosure of the object. He introduced the two algorithms planetarium and normal. One of the modern works are that of Vasquez-Gomes et al. and Kriegel et al. The first researchers use a technique called voxelization and ray casting. The quality of the model is determined by a utility function. The work of Kriegel et al. is based on the methods probabilistic voxel space and triangle mesh. Their modern work is challenged by Border, which outperformed their results with its NBV algorithm called SEE. The research question supported by relevant sub-questions are presented in Section 4.8.

5
NBV Simulation for LES

The objective is to find the most optimum NBV algorithm for the LES by focusing on the strategy (Section 5.1), consisting of the simulation setup (5.1.1), NBV algorithm code outline (5.1.2), boundary detection via triangulation (5.1.3), normal vector determination (5.1.4), candidate point calculation (5.1.5) and the termination criteria (5.1.6). Section 5.2 explains the six NBV selection methods and Section 5.3 presents the NBV simulation. Section 5.5 is a summary of this Chapter.

5.1. NBV strategy

The NBV algorithm SEE focuses on the boundary of a point cloud to perform a NBV analysis. This algorithm is used as a basis for the simulation of the LES. The SEE produces candidate points and selects the point with the minimum distance to the sensor as the NBV. Five more selection methods are added in this thesis to compare the mutual scan performance. The computer used for calculation has a the Central Processing Unit (CPU) i5-10400 2.90 GHz and 16 GB of Random Access Memory (RAM).

5.1.1. Simulation setup

The output of the sensor of LES is a set of discrete points, being a point cloud. The simulated objects are a sphere (5.150 points), a cone (10.000 points), a cube (23.205 points) and a plate (1.681 points). The major difference between the objects is that a sphere has a depth and different curvature values. It is also possible to obtain depth values from a cone, however the distance and curvature values can be similar. A cube and plate has no depth at initial scan nor a curvature value. Also obtaining the normal at the edges of a cube and plate and simulating this without activating the points at the side require a new method and algorithm.

The sensor is simulated as a point with a working distance of 0.6 m and a depth range of 0.3 m. All the points which are within 0.9 m distance, will be captured. This is illustrated in Figure 5.1 for the sphere and in Figure 5.2 for the cone.

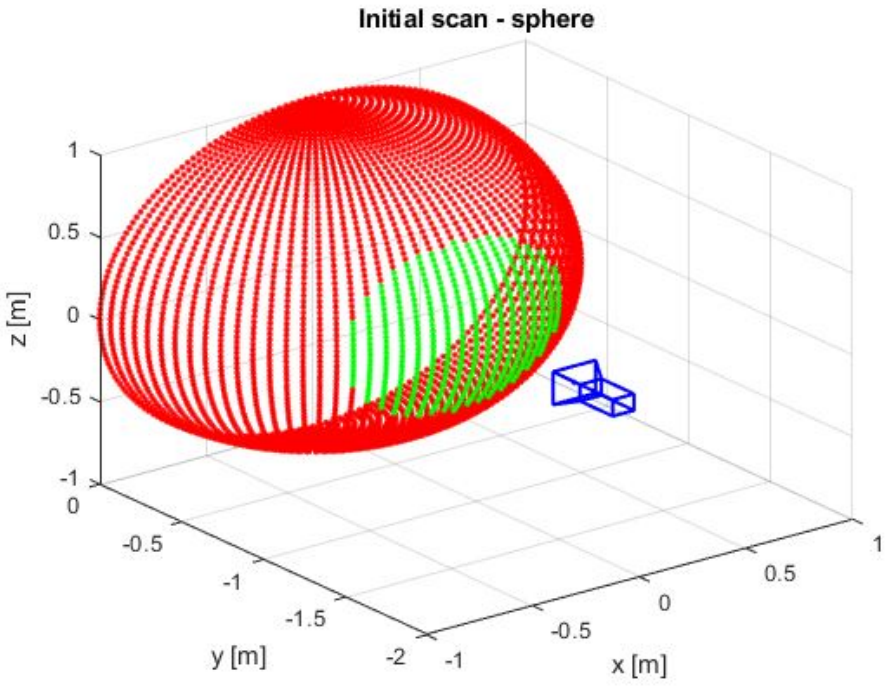


Figure 5.1: *Sensor position for initial scan - sphere*

The real sensor has a scan width of 0.4 m, however the simulation uses the value 0.9 m. This difference will not affect the comparison between the NBV selection methods and will result in uncluttered figures.

Compensating for this difference can be by changing the working distance of the sensor to higher values. Simulating the real scan width (0.4 m) and the working distance of the sensor (0.6 m) is a current limitation of MATLAB, which cause modification of the MATLAB built-in function for pairwise distance between two observed point sets (pdist2).

From NBV literature, the sensor is pointing towards the center of the object and an OBB is set, in this case 2 m wide and 2 m high, so the object is 1 m wide in the positive x direction and 1 m wide in the negative x direction. The same holds for the direction aligned with the z-axis. The object is sliced in half. so it only occupies 1 m in the negative y-direction. The simulated sensor is placed 0.6 m from the center (negative y-direction), making the initial lens position [0,−1.6, 0].

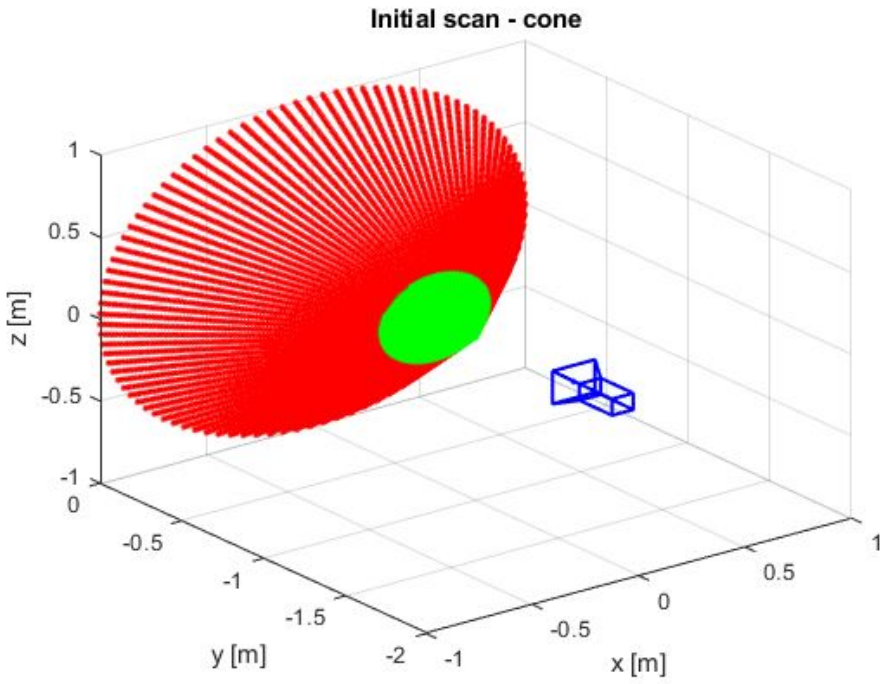


Figure 5.2: *Sensor position for initial scan - cone*

In order to ensure that the normal vector (Section 5.1.4) is always pointing to the correct direction, that is in the direction of the sensor, the environment is partitioned into four quadrants (5.3, being Q1 (upper right), Q2 (upper left), Q3 (lower left) and Q4 (lower right). That corresponds to the fact that all positive x- and positive z-values are in Q1, negative x- and positive z-values are in Q2, negative x- and negative z-values are in Q3 and positive x- and negative z-values are in Q4. This partitioning makes certain that the calculated and chosen NBVs, which are connected to normal vectors, are always outside the object, and not inside, which is incorrect and unreachable.

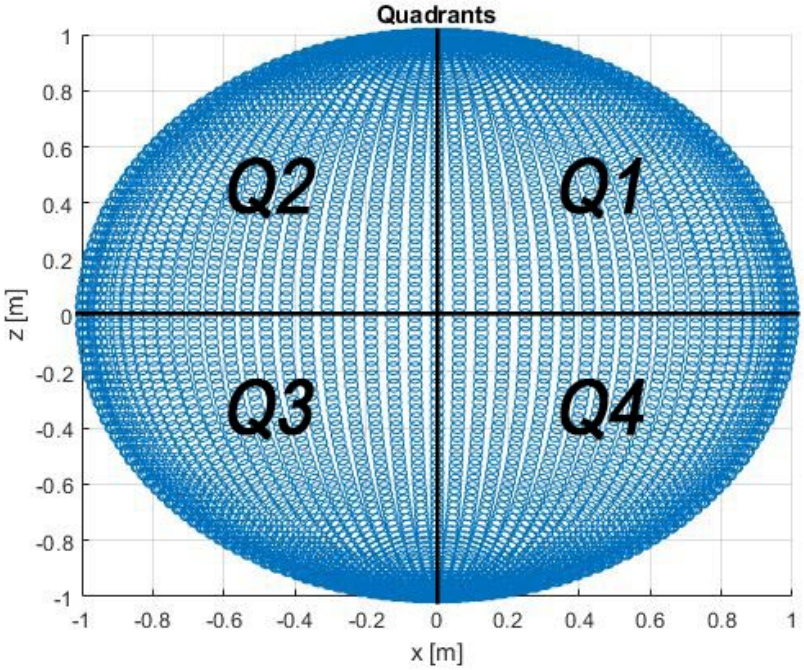


Figure 5.3: *Quadrants*

In order to make the NBV algorithm tunable for different methods and more important different objects, several simulation parameters are implemented (Table 5.1). The rho and sensed depth ($sense$) affects the number of activated points and thereby the scanned area. The number of neighboring points (ngb) is adjustable and is beneficial to use in the case of unstructured large point clouds. The ngb is for normal and curvature detection and will be adjustable to different kind of structured and unstructured point clouds. The correct Point of View (POV) is essential for the NBV determination process. The quality factor (Q) can count for quality by selecting more NBVs and increasing the chance of more surface coverage or many scans of a particular area. The max point to point distance ($dmax$) is used for triangulation. The distance between NBVs at multiple curvatures ($dista$) is a tuning factor mainly used at the curvature methods.

Table 5.1: *Simulation parameters*

Parameter	Value
ρ	0.6
$sense$	0.3
ngb	10
POV	[0, -1.6,0]
Q (sphere)	0 or 1
Q (cone)	5.5^{11}
$dmax$	1
$dista$ (sphere)	0.1 $[m]$
$dista$ (cone)	0.01597 $[m]$

5.1.2. NBV algorithm code outline

The simulated NBV algorithm consists of several steps. After initial scanning, the processes of partitioning in quadrants, boundary detection and normal and curvature calculation is performed, NBV selection and POV calculation starts. The three main codes are presented in the Appendices A, B and C.

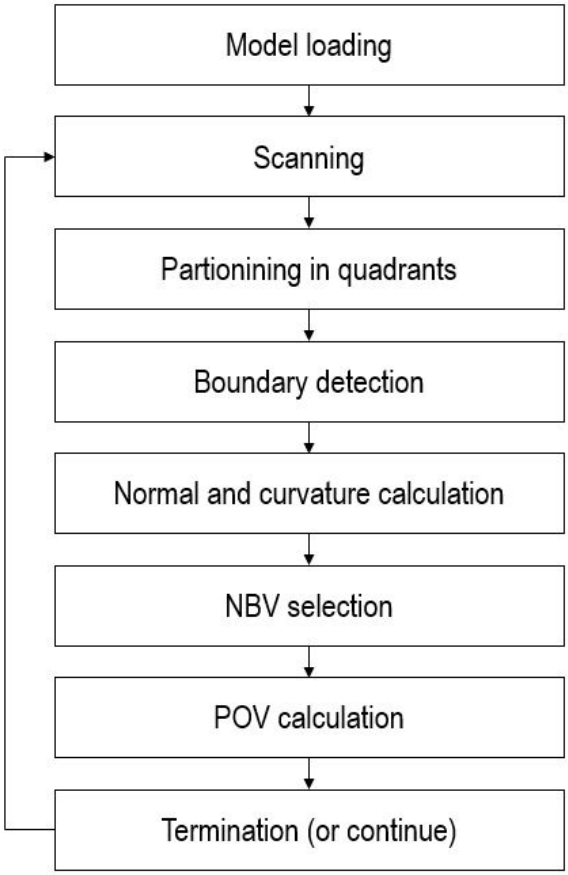


Figure 5.4: *NBV algorithm outline*

5.1.3. Boundary detection via triangulation

In order to select the NBV after the initial scan of an unknown object, the retrieved density of points needs to be analyzed. Local surface analysis is a crucial step in retrieving information about the obtained cloud, especially the boundary (frontier) points. These points are found by boundary detection and contain the edges and contour of the scanned segment. The boundary of a scanned segment is found by Delaunay triangulation [38]. The points, or in the discrete world named as vertices, are triangulated to check if an edge is a boundary. If this edge does not contain two different triangles in its surrounding, then it belongs to a boundary. Figure 5.5 is an illustration of the boundary detection of the front part of a sphere obtained by triangulation figure 5.25 illustrates the inner connection between the bounrdary points. Figure 5.7 further zooms in the edge area of the sphere to see the distinction between edge points and inner points based on neighbouring triangles.

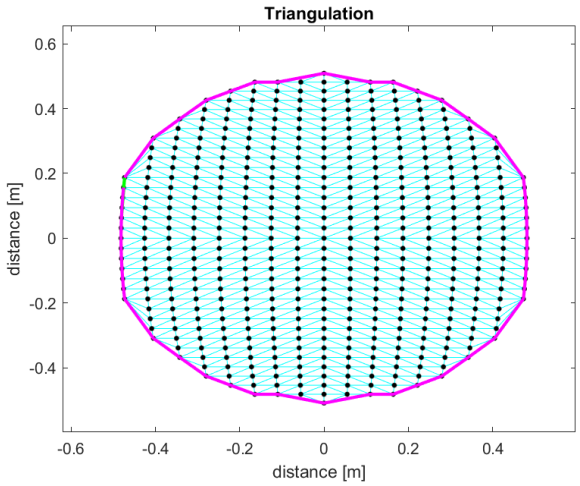


Figure 5.5: *Triangulation after first scan of the front part of a sphere*

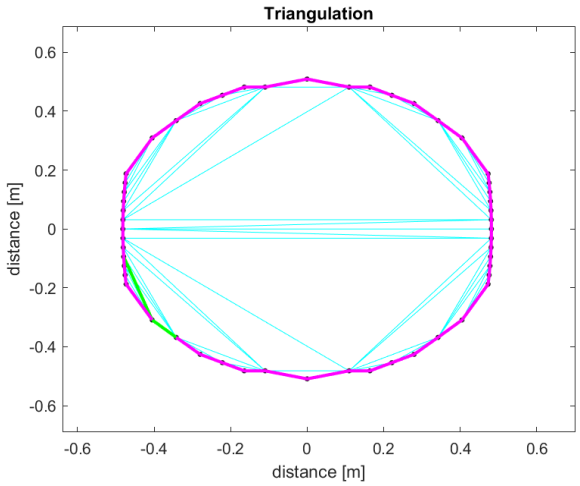


Figure 5.6: *Triangulation after first scan - interconnected boundary points*

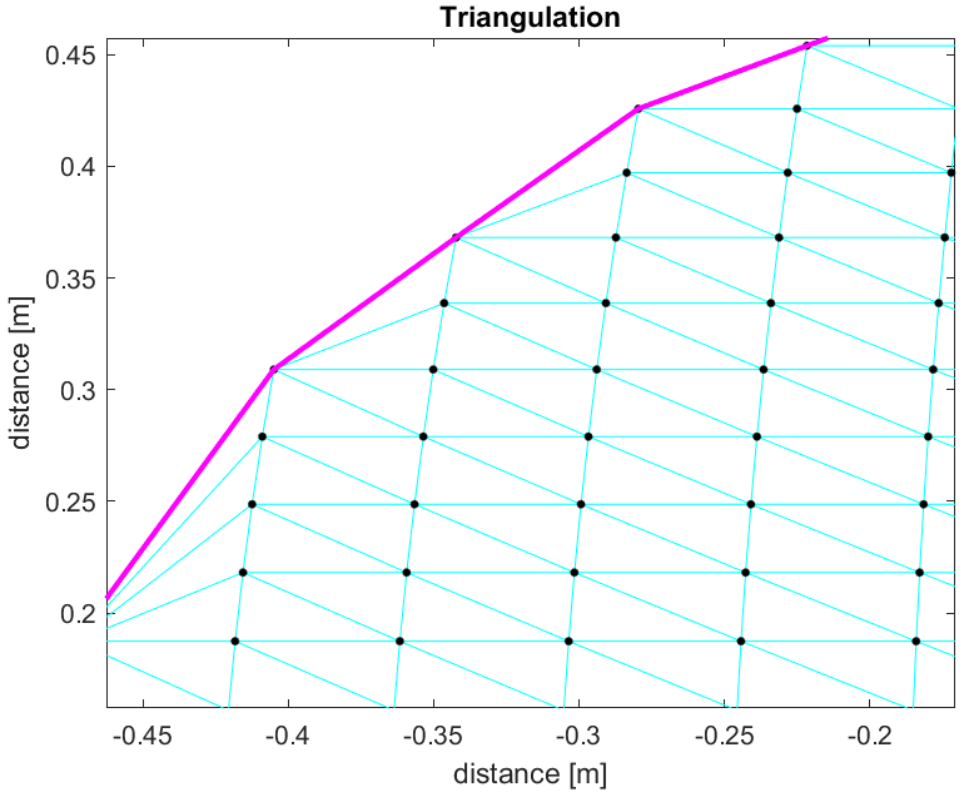


Figure 5.7: *Triangulation after first scan of the front part of a sphere - zoomed in [33]*

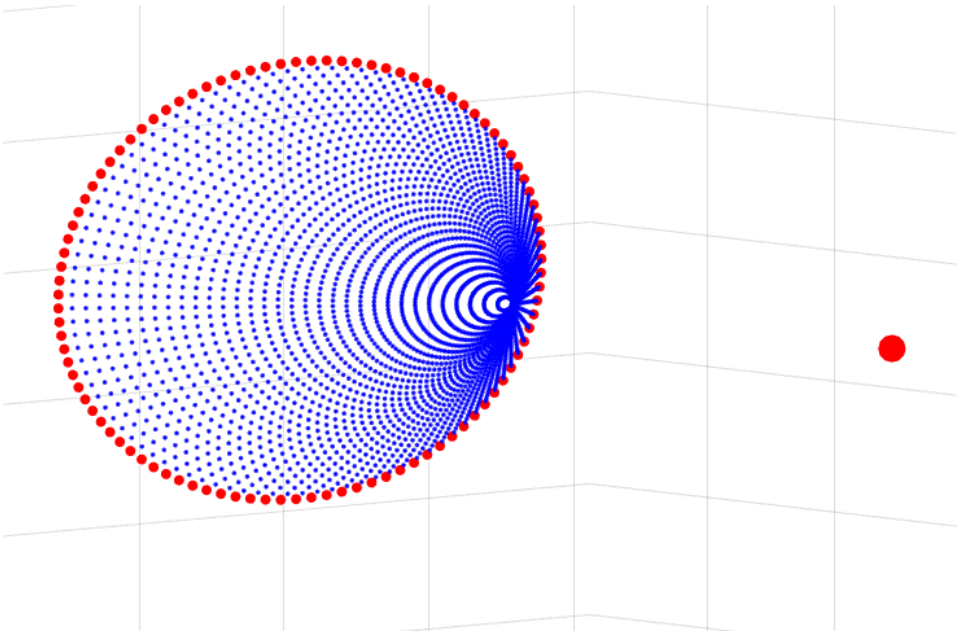


Figure 5.8: *Data segmentation of boundary points*

5.1.4. Normal vector determination

The frontier points can be separated and highlighted by the obtained data to perform calculations on them (Figure 5.8). These points are used for estimating the NBV by analyzing their normal vectors and their three components in the x-, y- and z-direction. For this, a nearest neighbor search method is performed to use the adjoining, or all the points within a distance, points for analysis of a particular area. The Kd-Tree structures data with space partitioning which is used to organize points in 3D. The number of neighboring points of the NBV algorithm in this thesis is set to nine om this thesis and is changeable. The normals are calculated by using a covariance matrix (C), where the p indicate the difference in position from neighboring points, being $p_i - p_k$.

The C is given by:

$$C = \begin{bmatrix} p_{11} & p_{12} & p_{13} \\ p_{12} & p_{22} & p_{23} \\ p_{13} & p_{23} & p_{33} \end{bmatrix}$$

An eigenvalue analysis results in the eigenvalue (ψ) and elevation angle (θ). The normals are the eigenvectors, $\Psi = \{\psi_1, \psi_2, \psi_3\}$, at the minimum eigenvector λ_{min}, solved by the eigenequation 5.1:

$$C\psi_i = \lambda_i \psi_i \tag{5.1}$$

The normals on the frontier points of the front part of a cone are depicted in Figure 5.9.

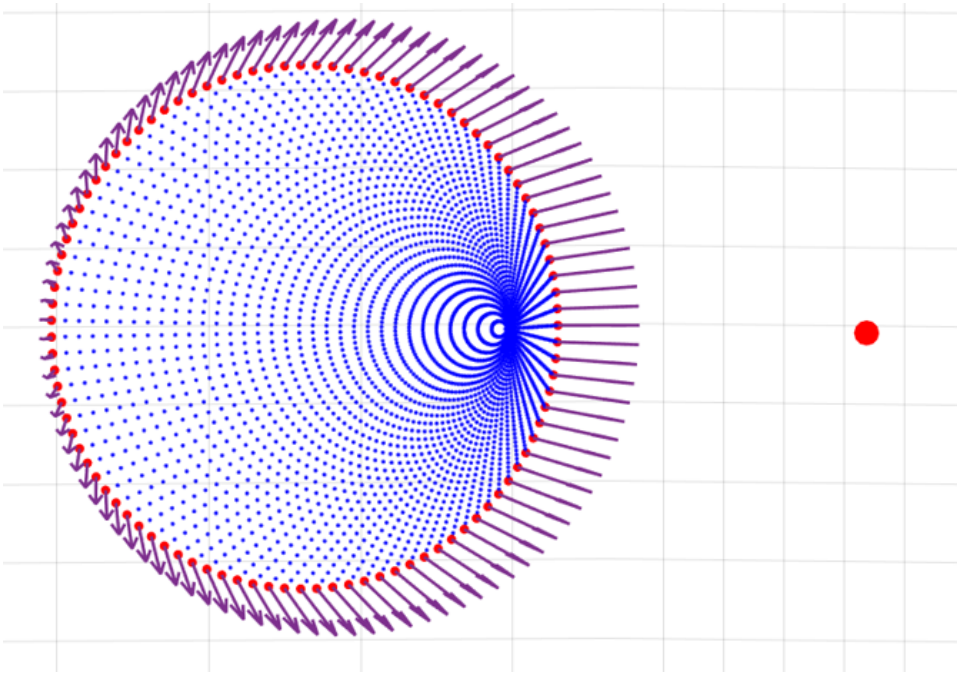


Figure 5.9: *The normals on the frontier points*

It is crucial to check if the normals are tangential to the frontier points (Figure 5.10).

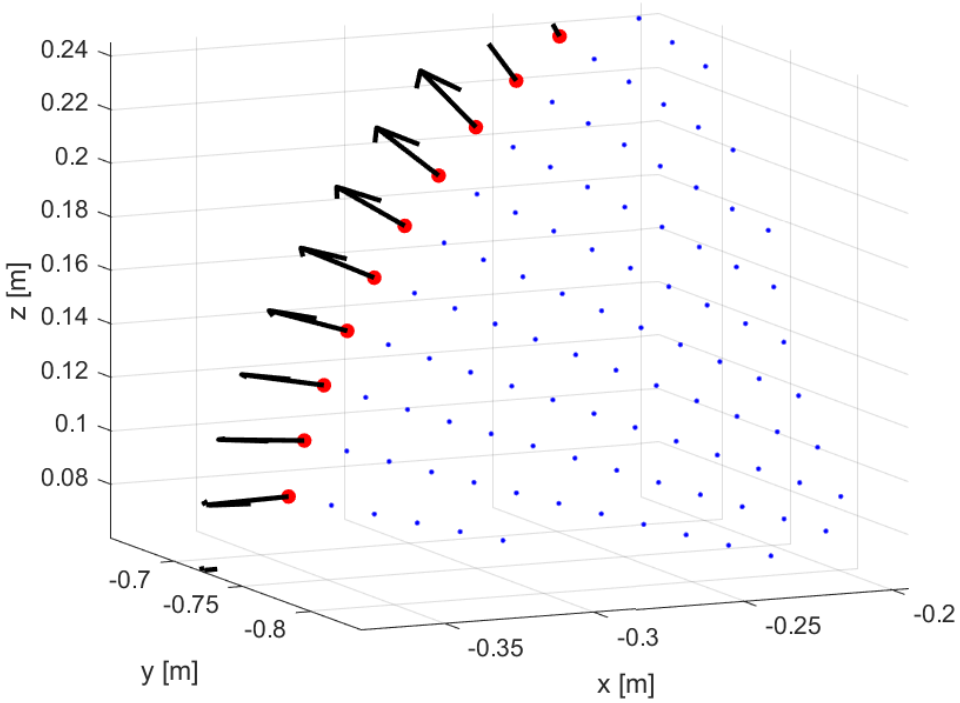


Figure 5.10: *Tangent check of the normals*

5.1.5. Candidate point calculation

To obtain the coordinates, the ϕ and θ are needed. The azimuth angle (Equation 5.2) and elevation angles (Equation 5.3) are calculated by using the normal vector component (n_i) at a point:

$$\phi = \arctan(\frac{n_y}{n_x}) \tag{5.2}$$

$$\theta = \arccos(\frac{n_z}{\sqrt{n_x^2 + n_y^2 + n_z^2}}) \tag{5.3}$$

These angles are used for obtaining the coordinates (Equations 5.4, 5.5 and 5.6) via the formula giving the relationship between Cartesian and spherical coordinates, including ρ:

$$x = \rho \sin\phi \cos\theta \tag{5.4}$$

$$y = \rho \sin\phi \sin\theta \tag{5.5}$$

$$z = \rho \sin\phi \tag{5.6}$$

The frontier points lead to the coordinates and angles of candidate points. A selection step (ranking) will determine the NBV(s). (Figure 5.11)

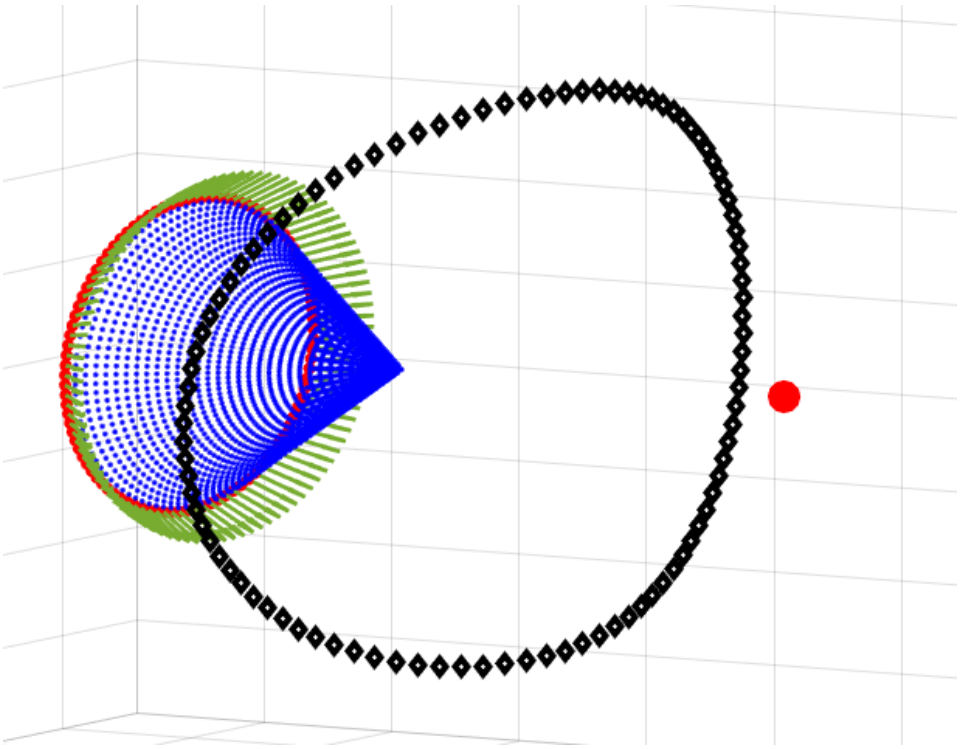


Figure 5.11: *Candidate points based on the normals at the boundary*

Another possibility is the estimation of the curvature at the frontier point, to be able to gain the most new information about the unscanned area.

The minimum eigenvalue corresponds to the least surface variance. The normal vector direction is set opposite to the sensor direction.

As for the curvature (κ) two widely used methods, being the Gaussian curvature ($\kappa = k_1 k_2$) or mean curvature (H_κ) ($H_\kappa = \frac{k_1+k_2}{2}$), where the principal curvatures, the first principal curvature (k_1) and second principal

curvature (k_2), are avoided as they are sensitive to sensor noise and are unable to calculate the curvature from a group of points and consequently require meshes [6].

Therefore, the curvature change is obtained by calculating the ratio between the minimum eigenvalue and the sum of the eigenvalues [39], given by Equation 5.7:

$$\kappa = \frac{\lambda_{min}}{\sum \lambda_n} \tag{5.7}$$

The surface coverage (S) is determined by dividing the total number of points of loaded object (P_{tot}) by the number of observed points (P_{obs}) during the simulation (Equation 5.8).

$$S = \frac{P_{tot}}{P_{obs}} \tag{5.8}$$

5.1.6. Termination criteria

The algorithm needs to stop sending the robots to other positions when a termination criterion is reached to stop the iterative calculation and sensor movement process. This condition is linked to the OBB and ρ. The algorithm is finished when a NBV location in x or z is reached which is the working sensor distance added to the boundary value. For stability, 0.01 m is deducted from this value. At a OBB of 1 and ρ of 0.6, this value becomes 1.59. If the exact value of 1.6 is chosen, the algorithm is not be able to stop when a NBV has the value 1.59999, which is often the case. For the constrained case the termination criterion is set to 0.8 (in the x- and z-direction). Also, in order to prevent that the algorithm falls in an endless loop, a maximum scan round count of seven is set. This is also to acquire the maximum results from the NBV selection methods. The algorithm self-terminates when one of these termination values are reached.

5.2. NBV selection

Ranking of the candidate viewpoints is the essential step in determining the NBV. Six NBV selections methods are described: (1) minimum distance, (2) maximum distance, (3) minimum and maximum distance, lowest curvature (4), lowest curvatures (5), which is tunable, and a hybrid version (6). The methods hold for objects with a depth, which a sphere and cone are. For a cube, which is a plate at initial scan, the selection of the maximum and minimum x- and z-coordinates is presented.

5.2.1. Minimum and maximum distance

Boundary points are used for generating NBV candidates. NBVs are selected based on their cumulative and origin distances [18]. The cumulative distance of a NBV is given by the difference between the current view position x_i and the new view position. The difference between the initial viewpoint x_0 and the new view position is the original distance. NBV selection is based on the minimization of the global distance from the set of candidate viewpoints (Equation 5.9).

$$v_{i+1} = \arg min(||x - x_0||) \tag{5.9}$$

In case of the absence of a new viewpoint, the NBV is selected which minimizes the local distance, Equation 5.10:

$$v_{i+1} = \arg min(||x - x_i||) \tag{5.10}$$

The ranking of NBVs based on minimum distance and their corresponding graph (distance over point/index number) is shown in Figure 5.12. An offset of ten percent is added to the minimum distance value in order to capture other minimum values which are the same or nearby. The mean of these value is selected in the case when there are four candidates on the right and four on the left (green diamonds). The red point is the NBV

including its azimuth and elevation angle. The light blue points is the point cloud segment obtained from that NBV.

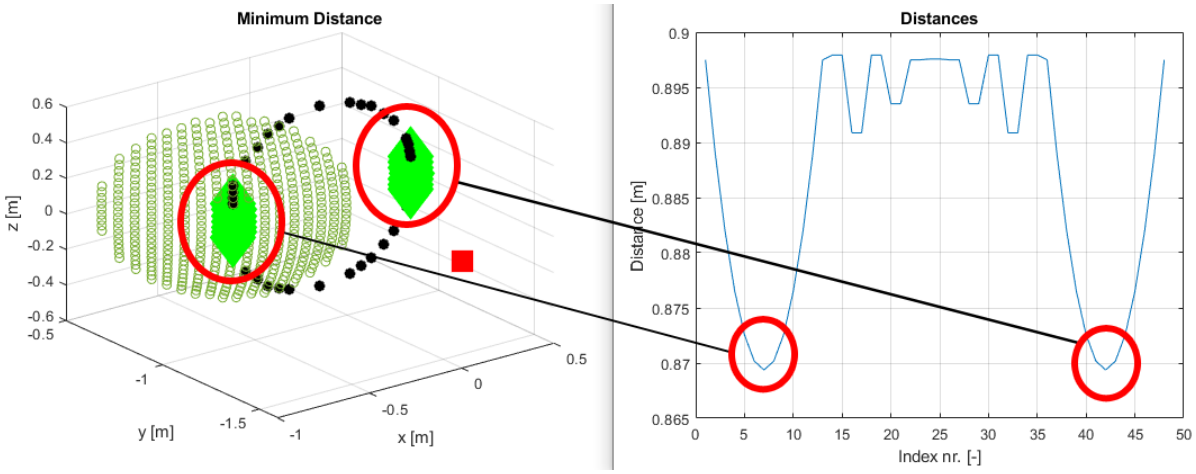


Figure 5.12: *Minimum distance detection*

The same process is done for the maximum distance from the sensor to determine the difference with the previous method. The third method is the combination of the minimum and maximum distances to determine the difference in total amount of scans, surface coverage, computation time and distance.

5.2.2. Lowest curvature(s)

Another method to rank the NBVs is by analyzing the points with lowest curvature, called a seed point [6]. For every seed point, neighboring points are found. The normal of every neighbor point is compared to the normal of the seed point. A threshold angle value α_{th} determines if the seed point, after the working distance in the correct direction, can be used as NBV. The minimum curvature value is selected. An offset is added to find the NBVs which have similar values or are nearny. In Figure 5.13 one NBV is detected with the help of the minimum value in the graph in Figure 5.14.

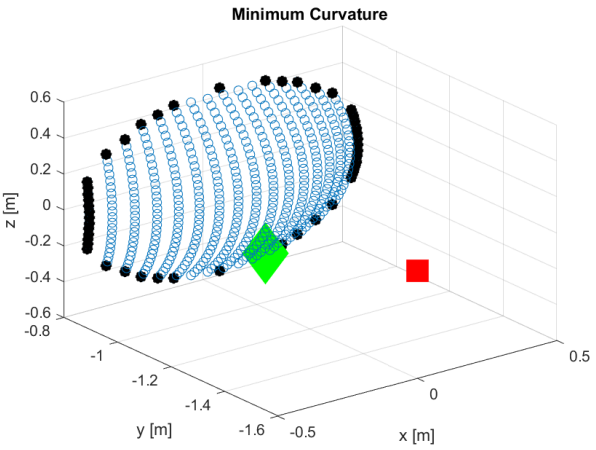


Figure 5.13: *Minimum curvature detection*

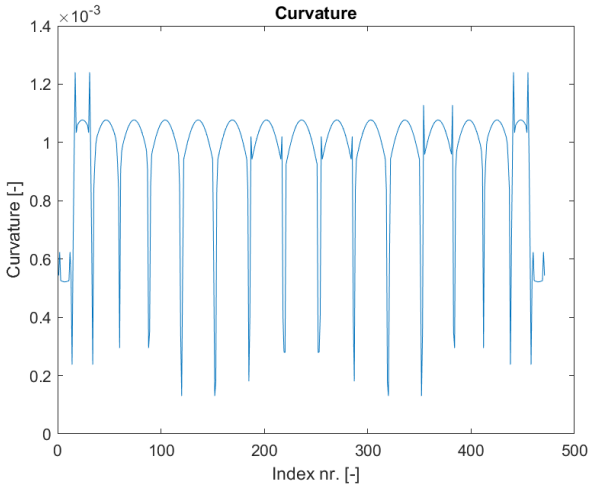


Figure 5.14: *Curvature graph*

As for the fifth NBV selection method, using the Q results in multiple lower curvature values. In Figure 5.15 and Figure 5.16 more (eight) NBVs are found and from that the partially build model is shown after scanning from these eight selected new coordinates, including their ϕ and θ. Setting Q to a very high value activates all the boundary points as the NBV, resulting in ultra high scan quality at the cost of a great many number of

scans. This is illustrated in Figure 5.17 and 5.18. For optimization, $dista$ can be set to a value to define the distance difference between NBVs.

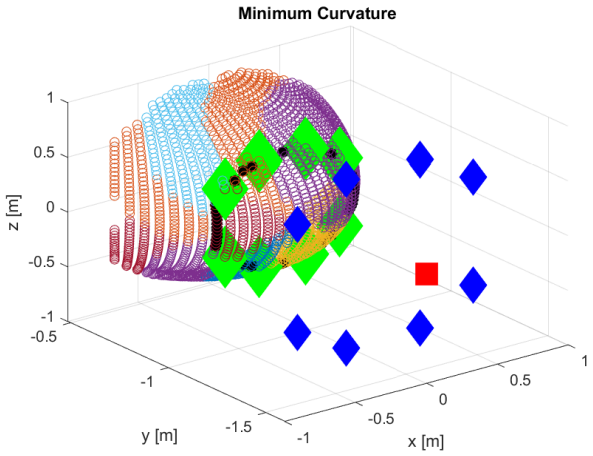


Figure 5.15: *Minimum curvature detection with offset value*

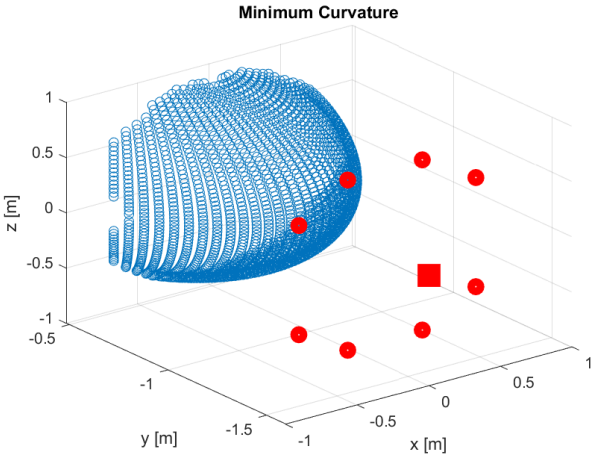


Figure 5.16: *Minimum curvature detection with offset value (clean view)*

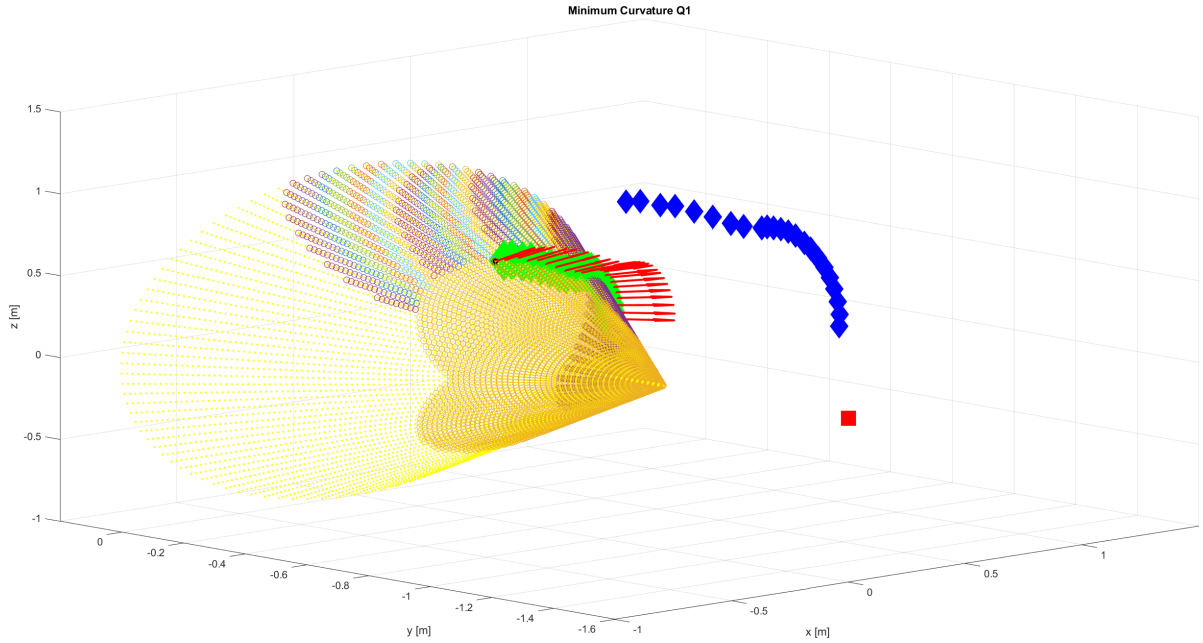


Figure 5.17: *Selection of all border points as NBVs (angled view)*

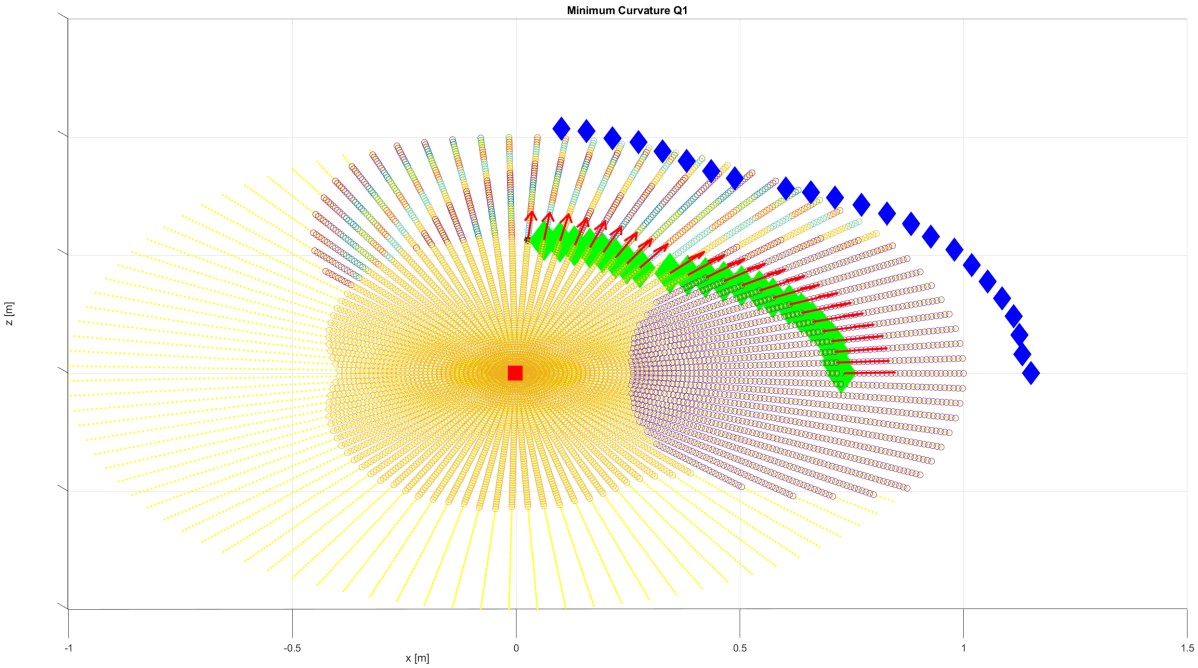


Figure 5.18: *Selection of all border points as NBVs (front view)*

5.2.3. Hybrid version

A hybrid version consisting of the single minimum distance and minimum curvature value is also tested to determine the difference and performance of the methods.

5.2.4. NBV for cubical objects or plates

After obtaining the first segment of a point cloud after an initial scan, the outmost coordinates can be selected in the x- and z-direction by using the max and min function of MATLAB. This gives, going anticlockwise, the most right, upper, left and lower point. These four points are the edges. It is not possible to obtain a curvature values as a cube and plate does not have any curvature at the front side, except at the corner where the other side starts.

5.3. NBV simulation

This Section explains the NBV calculations and selections for the first two siumulated scan rounds. The hybrid method for the sphere is described in detail.

After the initial scan, border detection is performed to extract the border points for further NBV evaluation. This is visualized in Figure 5.19 and is also used to check if the triangulation step is performed correctly and no different points except the border ones are selected. The triangulation figures were presented previously in Section 5.1.3.

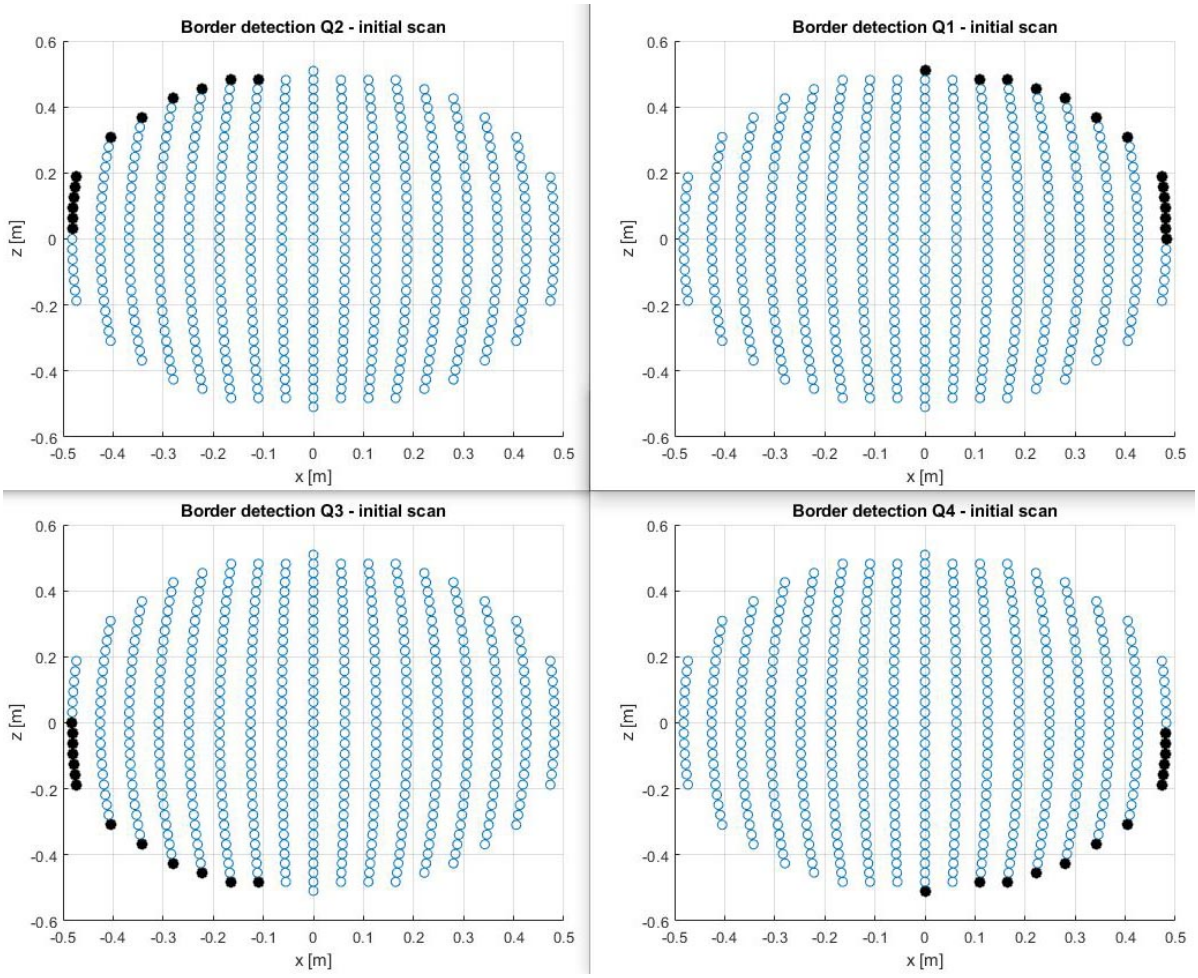


Figure 5.19: *Border detection per quadrant - initial scan*

In the following step, NBV determination is performed based on the chosen selection method. The hybrid version selects the minimum value of the calculated distances and curvatures. The graphs of these calculations are given in Figure 5.20 and 5.21. The index number corresponds with the location of the border points.

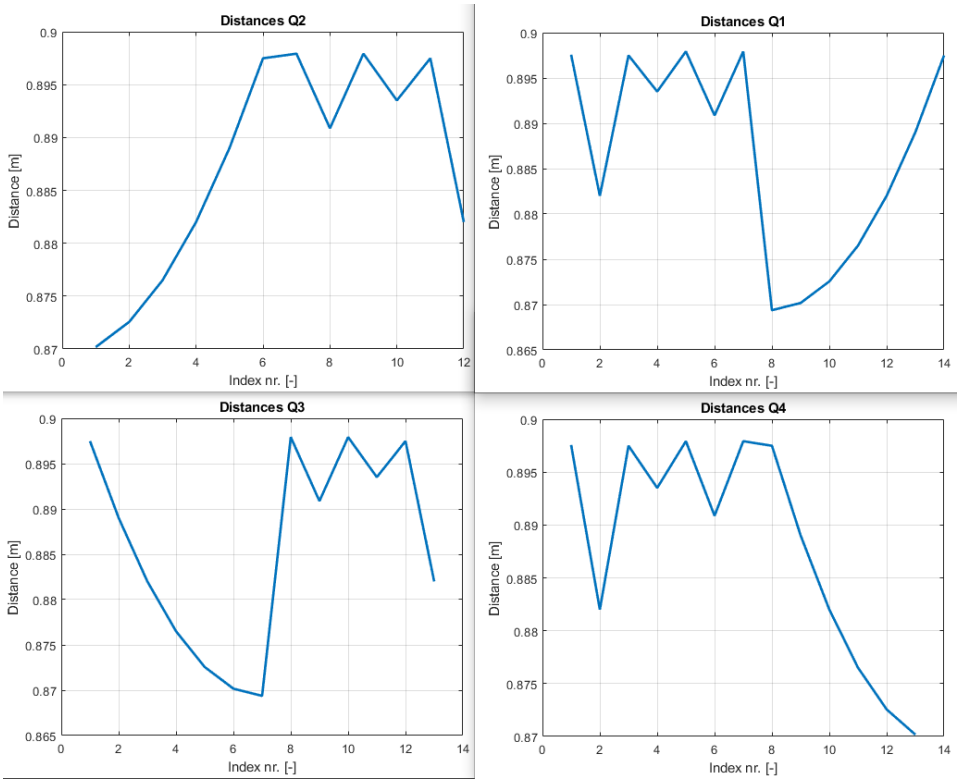


Figure 5.20: *Distances from sensor location per quadrant after initial scan*

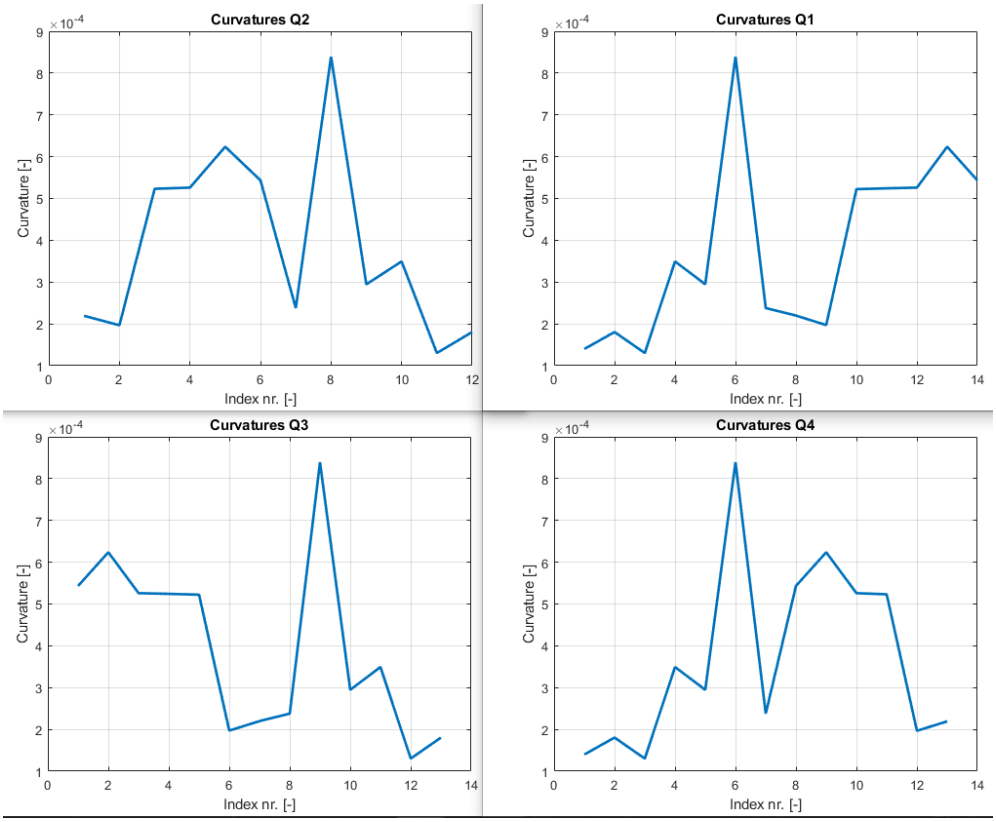


Figure 5.21: *Curvatures per quadrant after initial scan*

The minimum values results in the selection of the following NBVs (Figure 5.22), where the green diamonds

are the selected NBV borders and the blue diamonds are the NBV coordinates. The red arrows are the normal vectors and their direction. The red square is the initial sensor position. The same procedure continues for Q2, Q3 and Q4. The result of NBV determination from the first scan round is depicted in Figure 5.23.

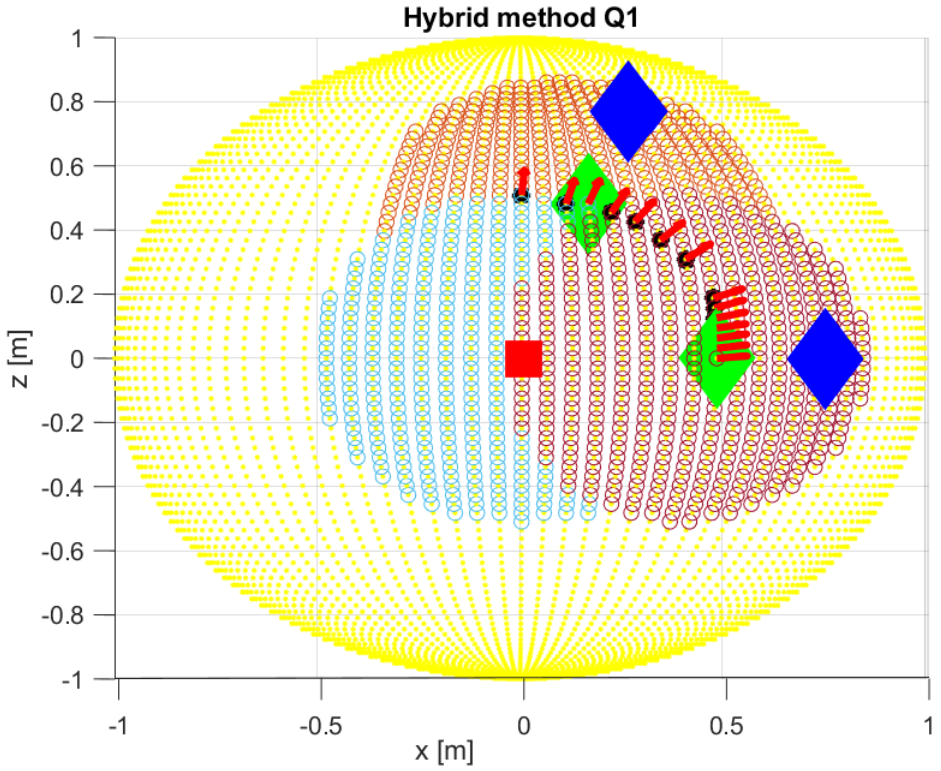


Figure 5.22: *NBV selection at Q1*

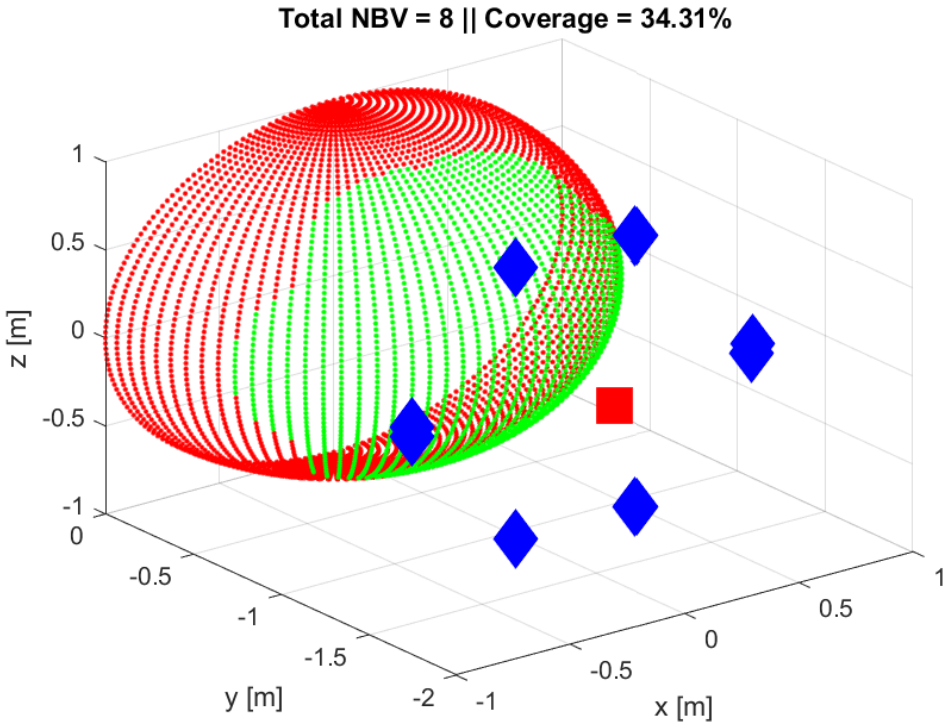


Figure 5.23: *Scan result first NBVs*

The algorithm continues with the new obtained coordinates. Again the border points (Figure 5.24) are determined via triangulation (Figure 5.25), as also their normals. The minimum distance and curvature values are calculated and the NBVs are selected (Figure 5.26) corresponding to these minimum values.

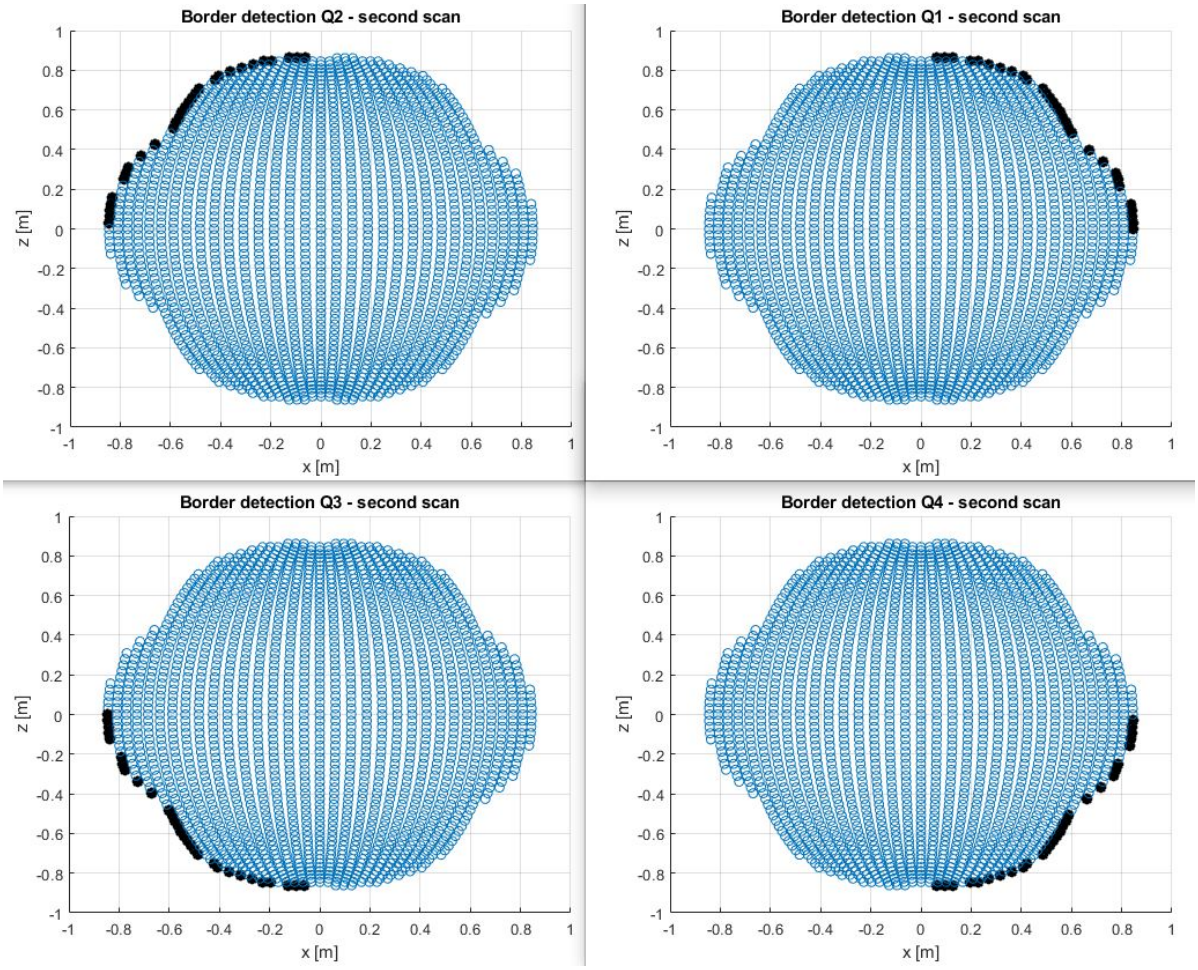


Figure 5.24: *Border detection per quadrant - second scan*

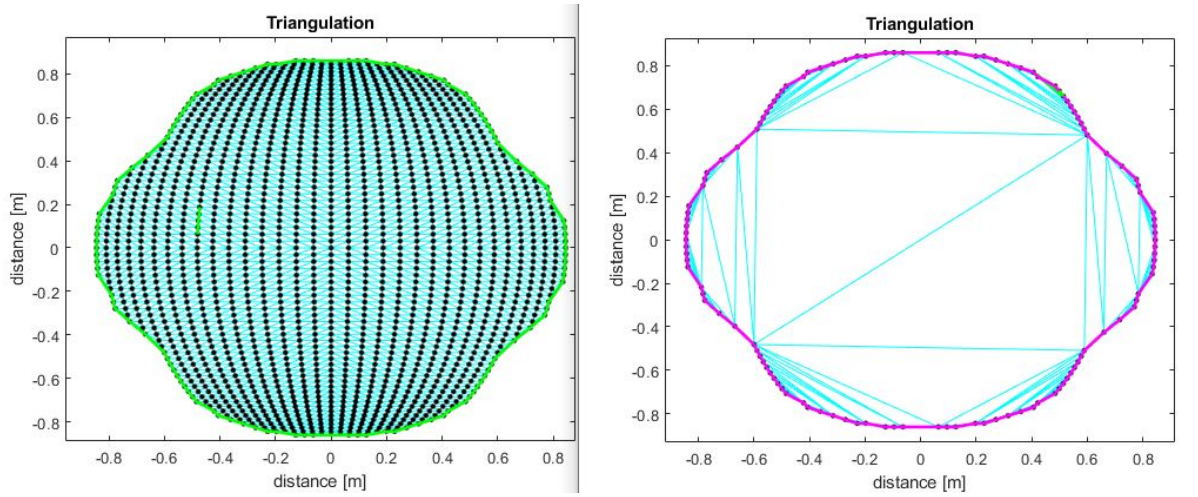


Figure 5.25: *Triangulation during second scan*

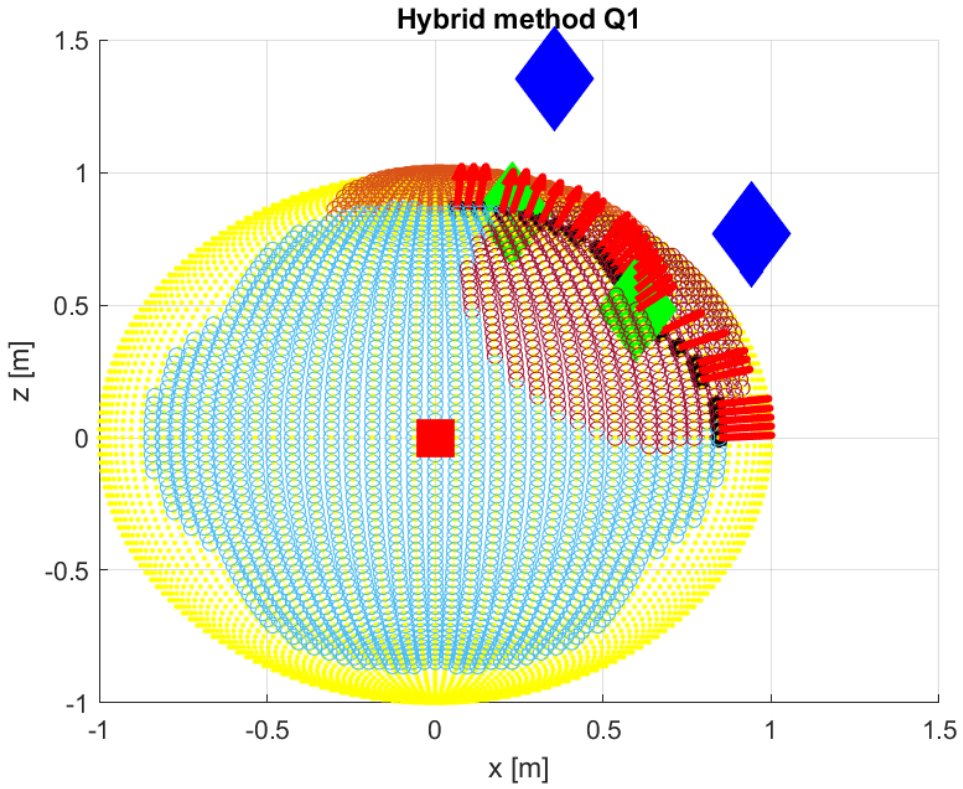


Figure 5.26: *NBV selection at Q1 - second scan*

The result of NBV determination from the second scan round is depicted in Figure 5.27.

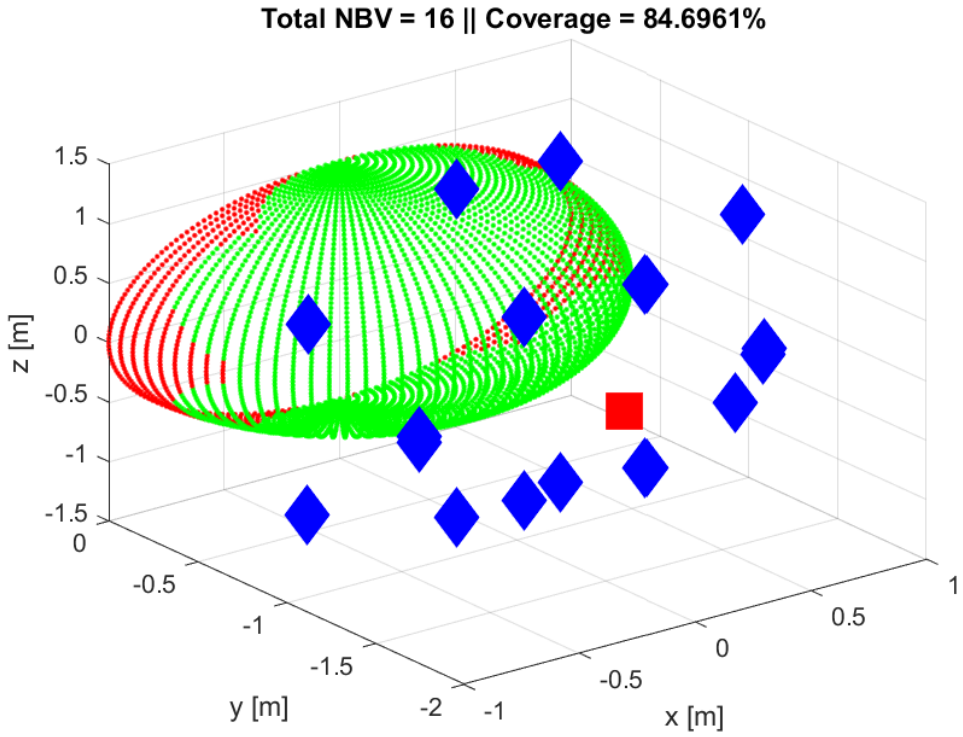


Figure 5.27: *Scan result second set of NBVs*

This process continues until one of the termination criteria is met. The total distance covered by the robotic arm is calculated via the Euclidean length between the first POV, the first NBV and all the successive NBVs.

5.4. Summary of Chapter 5

This Chapter explained the NBV strategy, ranking methods and the simulated LES setup. The boundary detection of an obtained point cloud is done by Delaunay triangulation. The boundary points are used for normal and curvature calculations. NBV ranking is done by the methods of minimum and maximum distances, lowest curvature, multiple low curvatures and a hybrid version. The NBV algorithm has two termination criteria. The NBV simulation takes place in several scan rounds, where the obtained NBV locations are ordered anti-clockwise.

6
Results and Discussion

This Chapter presents the results of the simulated six NBV selection methods for the two objects sphere (Section 6.1) and cone (Section 6.2) for the unconstrained case. The distance (between the POV and candidate points) and curvature are the two main comparison methods. The cube or plate has, due the absence of curvature, only one NBV selection method, being the coordinate information, and is prone to future research (Section 6.3). The following Section, 6.4, is a second simulation for the constrained case. Also, the results are discussed in this Chapter.

6.1. NBV results for a spherical object in unconstrained case

The simulation resulted in different solutions. These are presented in Table 6.1. The minimum and maximum distance combined method and the hybrid method resulted in full surface coverage. The first method required four more scans (34) and more robot movement distance (46.6 m) than the hybrid version (100 percent in 30 scans and 75.4 s), however the computational time was slightly less (73.6 s). The hybrid method terminated more optimally (in 2 scan rounds after the initial scan). The combined minimum and maximum distance method needed one more scan round (3 in total). The minimum distance method also resulted in nearly full coverage (99.96 percent), however the computation time was very high (130.9 s) as it did not terminate in the first scan round and waited to reach the predefined maximum allowable scan rounds. The lowest surface coverage is obtained with the minimum curvature method (69.9 percent). The lowest curvatures method required only 12 scans, however the surface coverage stayed around 80 percent. The minimum distance and less sensor movement distance (41 m) than the hybrid method. The hybrid method covered the object in 30 scans and took 83.5 seconds to calculate and needed to move the robotic arm 44.8 m in total. The maximum distance method has the lowest computational time and sensor movement distance, however it only scans 74.2 percent of the sphere. The minimum curvature method results in the lowest surface coverage of 69.9 percent.

Table 6.1: *NBV results for the sphere*

Method	**Scans [-]**	**Coverage [%]**	**Time [s]**	**Total distance [m]**	**Termination round**
Minimum Distance	28	99.96	130.9	43.1	7
Maximum Distance	20	93.8	83.8	33.2	4
Min+Max Distance	34	100	73.6	46.6	3
Minimum Curvature	14	69.9	44.7	19	2
Minimum Curvatures	12	80	113.6	14.8	7
Hybrid	30	100	75.4	44.8	2

Figure 6.1 illustrates the model build up plotted against the number of scans. The hybrid method covers, compared to minimum and maximum distance method, more area in fewer scans. The minimum distance method requires several scan rounds to achieve nearly full coverage. (Figures 6.2 and 6.3).

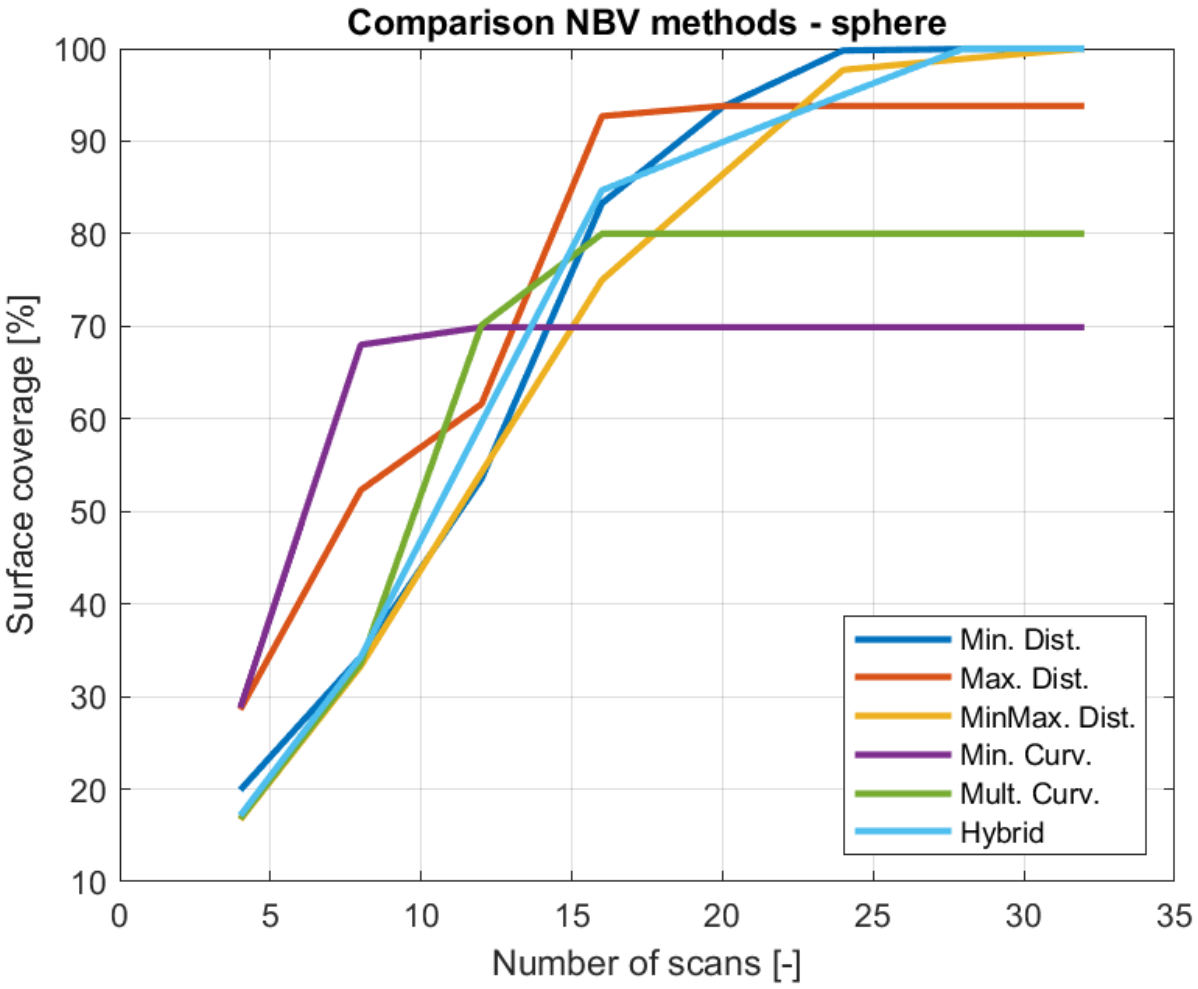


Figure 6.1: *NBV comparison - sphere*

The minimum distance method for NBV selection resulted in a surface coverage of 99,96 percent in a total of 28 scans, however it results in small holes at the both sides of the object, which is the missing 0.04 percent of coverage. This is depicted from two different angles in Figure 6.2 and Figure 6.3, where the blue diamonds are the NBV locations.

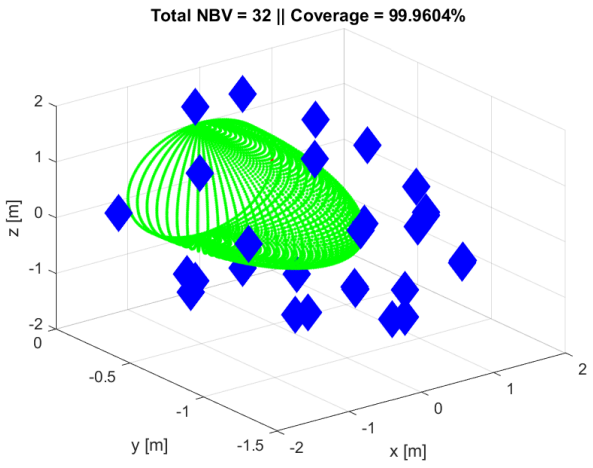


Figure 6.2: *Simulation result - minimum distance - sphere - angled view*

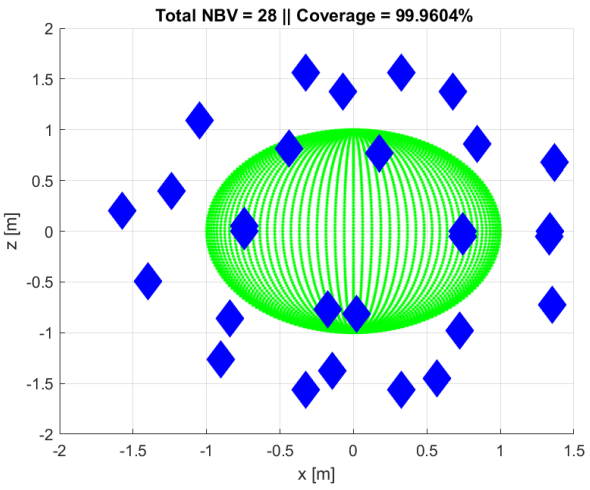


Figure 6.3: *Simulation result - minimum distance - sphere - front view*

The maximum distance method results in unacceptable holes in the upper and lower side of the object. Also part of the left and right side of the object remain unscanned. Figures 6.4 illustrates this from an angled view and Figure 6.5 is a front view.

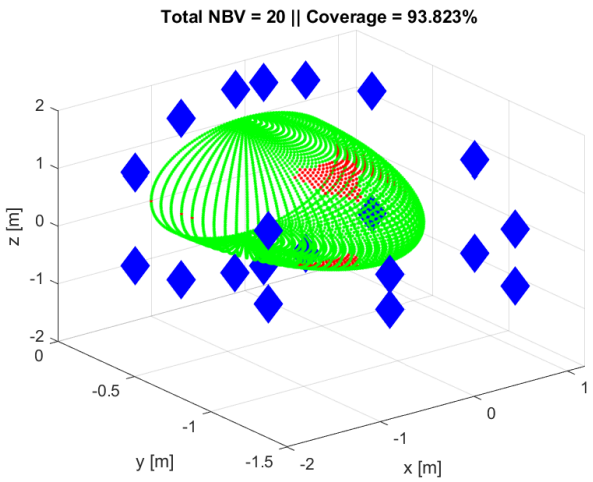


Figure 6.4: *Simulation result - maximum distance - sphere - angled view*

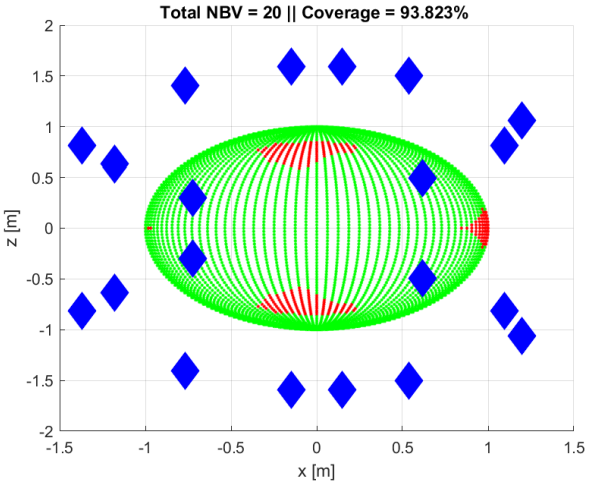


Figure 6.5: *Simulation result - maximum distance - sphere - front view*

The minimum and maximum distance method covers the entire object. There are NBV positions which are near each other (Figures 6.6 and 6.7).

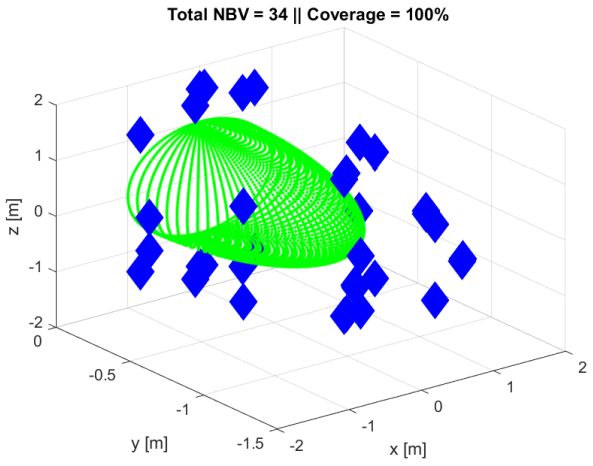


Figure 6.6: *Simulation result - minimum and maximum distance - sphere - angled view*

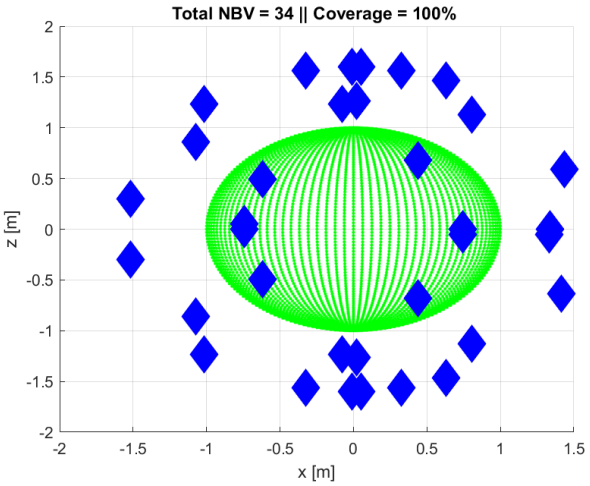


Figure 6.7: *Simulation result - minimum and maximum distance - sphere - front view*

The minimum curvature method dominantly moves in the z-direction. Therefore the x-axis is not visited and remains unscanned. Final NBV locations are close to each other (Figures 6.8 and 6.9).

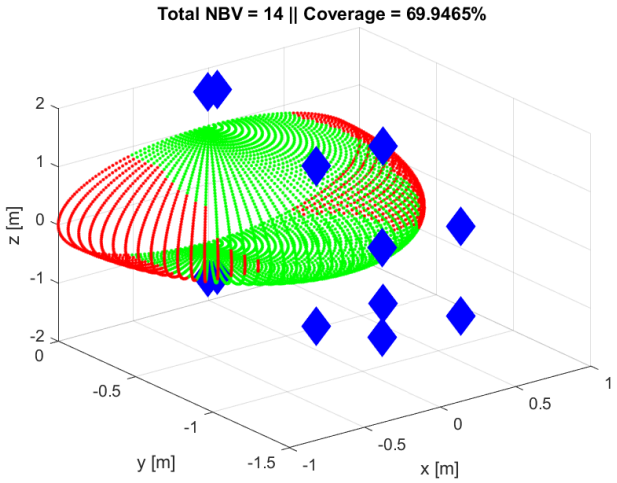


Figure 6.8: *Simulation result - minimum curvature - sphere - angled view*

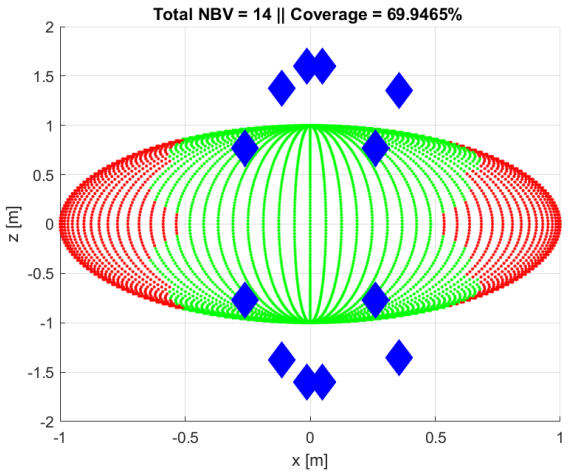


Figure 6.9: *Simulation result - minimum curvature - sphere - front view*

The multiple curvatures method also dominantly moves in the z-direction. Therefore large parts of the x-axis remain unvisited. (Figures 6.10 and 6.11).

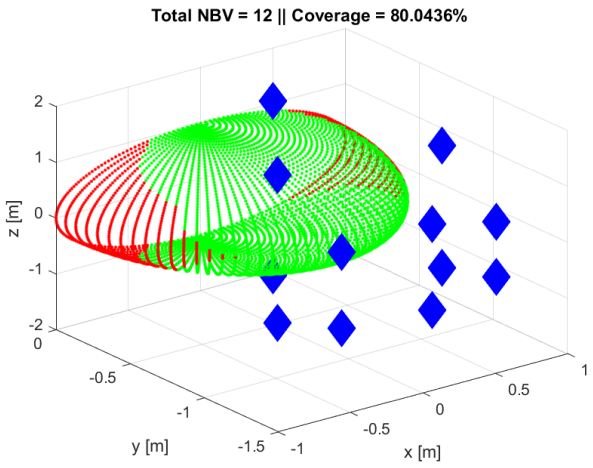


Figure 6.10: *Simulation result - multiple curvatures - sphere - angled view*

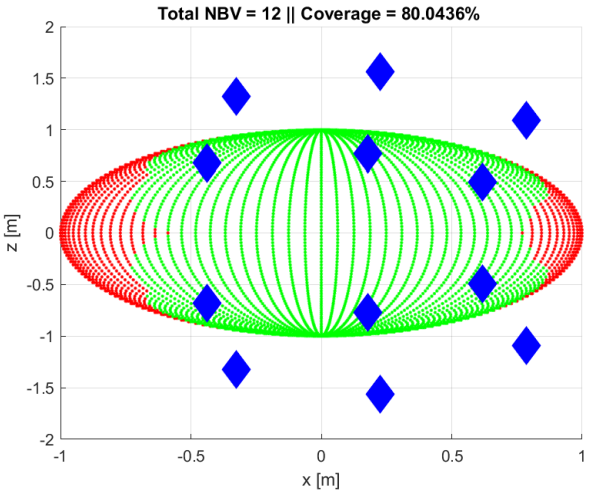


Figure 6.11: *Simulation result - multiple curvatures - sphere - front view*

It is possible to activate all the border points which will result in a major amount of scans (544 in total), however still not full coverage is achieved with this method and two percent (at the sides) of coverage is missed. (Figure 6.12 and Figure 6.13).

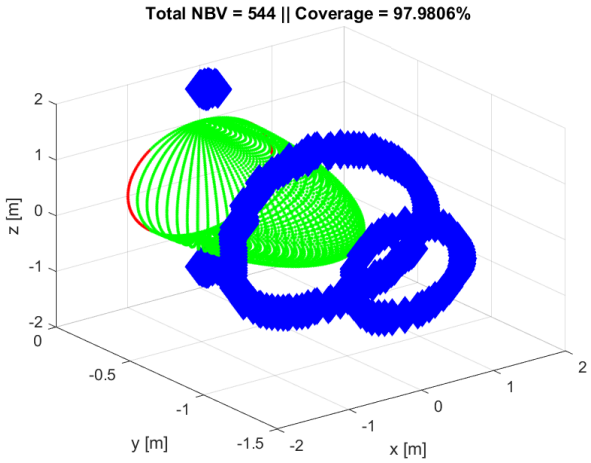


Figure 6.12: *Simulation result - all curvatures - sphere - angled view*

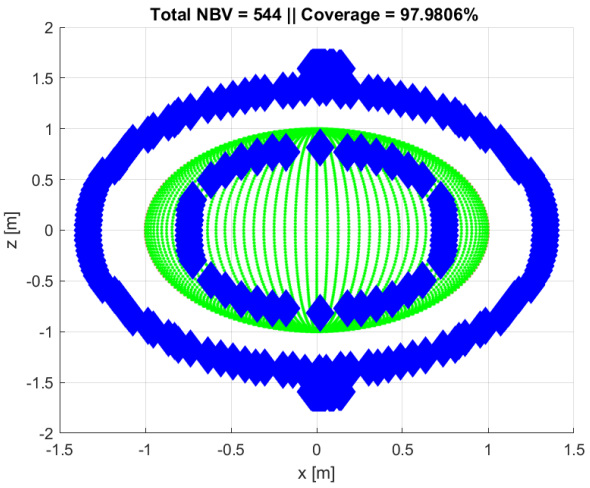


Figure 6.13: *Simulation result - all curvatures - sphere - front view*

The hybrid method covers the entire object. Except for the final scan round, the NBVs are located at acceptable distances and are not close too each other, pointing to more stable results (Figures 6.14 and 6.15).

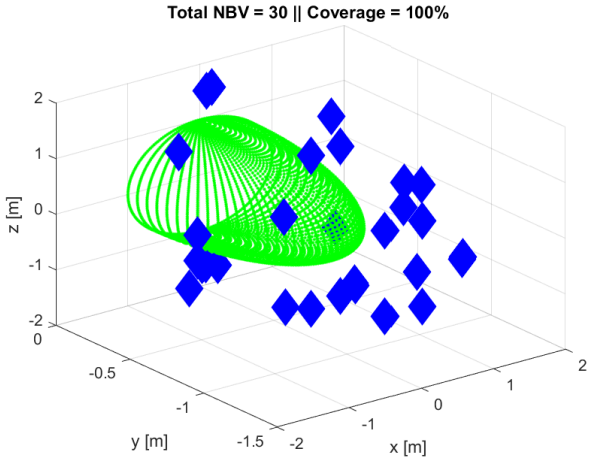


Figure 6.14: *Simulation result - hybrid - sphere - angled view*

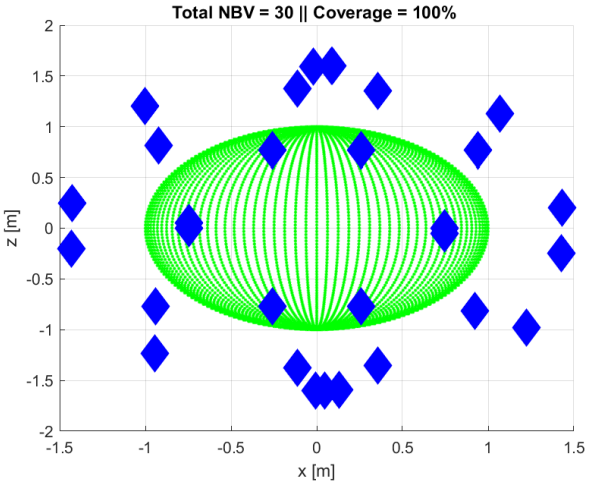


Figure 6.15: *Simulation result - hybrid - sphere - front view*

The NBV coordinates, the angles ϕ and θ and the scan sequence from the hybrid method for the sphere are tabulated in Table 6.2. The scanning takes place in rounds and NBV positions are selected anti-clockwise (from Q1 to Q2, Q2 to Q3 and Q3 to Q4). The NBV results of the other five methods are given in Appendix D.

Table 6.2: *NBV coordinates and angles - hybrid method - sphere*

Scan [-]	**x** [m]	**y** [m]	**z** [m]	ϕ [°]	θ [°]
1	0.2605	-1.3777	0.7708	10.5564	28.8000
2	0.7453	-1.4153	0.0000	26.0538	0
3	-0.2605	-1.3777	0.7708	-10.5564	28.8000
4	-0.7449	-1.4146	0.0503	-26.0543	1.8000
5	-0.2605	-1.3777	-0.7708	-10.5564	-28.8000
6	-0.7453	-1.4153	0.0000	-26.0538	0
7	0.2605	-1.3777	-0.7708	10.5564	-28.8000
8	0.7449	-1.4146	-0.0503	26.0543	-1.8000
9	0.3551	-0.7802	1.3509	23.2660	57.6000
10	0.9396	-1.0401	0.7708	40.2527	28.8000
11	-0.1137	-0.8063	1.3772	-9.4033	59.4000
12	-0.9227	-1.0218	0.8145	-40.2130	30.6000
13	-0.1137	-0.8063	-1.3772	-9.4033	-59.4000
14	-0.9396	-1.0401	-0.7708	-40.2527	-28.8000
15	0.3551	-0.7802	-1.3509	23.2660	-57.6000
16	0.9227	-1.0218	-0.8145	40.2130	-30.6000
17	0.0885	-0.0369	1.5968	90.0000	86.4000
18	1.4322	-0.6845	0.2005	63.8809	7.2000
19	-0.0197	-0.0553	1.5929	90.0000	84.6000
20	-1.4258	-0.6815	0.2503	-63.8763	9.0000
21	-0.0066	-0.0185	-1.5992	90.0000	-88.2000
22	-1.4322	-0.6845	-0.2005	-63.8809	-7.2000
23	0.0443	-0.0185	-1.5992	90.0000	-88.2000
24	1.4258	-0.6815	-0.2503	63.8763	-9.0000
25	0.0907	-0.0336	1.5968	90.0000	86.4000
26	1.0698	-0.3673	1.1314	69.4766	45.0000
27	-1.0001	-0.3446	1.2002	-69.2946	48.6000
28	-0.9433	-0.3874	-1.2328	-66.4633	-50.4000
29	0.1326	-0.0553	-1.5929	90.0000	-84.6000
30	1.2227	-0.3213	-0.9807	74.7402	-37.8000

6.2. NBV results for a conical object in unconstrained case

The six NBV methods also resulted in different solutions for the cone (Table 6.3). Now, the minimum distance method, together with maximum distance combined method and the hybrid method, resulted in full surface coverage (23 scans versus 30 scans versus 21 scans). The hybrid method required far less computation time (68.8 *s*) than the minimum distance and minimum and maximum distanec method and reached the termination criterion in two scan rounds (after the initial scan). The minimum curvatures method has the lowest computational time and sensor movement distance, however it only scans 76.4 percent of the cone. The maximum distance method asked for the most time (169.8 *s*) and gave back 96.1 percent. The minimum distance, maximum distance, minimum and maximum distance and minimum curvatures methods terminated at the predefined maximum allowable scan round, which is undesirable.

Table 6.3: *NBV results for the cone*

Method	Scans [-]	Coverage [%]	Time [s]	Total distance [m]	Termination round
Minimum Distance	23	100	157.2	32.4	7
Maximum Distance	19	96.1	169.8	31.4	7
Min+Max Distance	30	100	154.8	32.3	7
Minimum Curvature	15	90.9	57	18	2
Minimum Curvatures	13	76.4	45.4	7.5	7
Hybrid	21	100	68.8	32.1	2

Figure 6.16 illustrates the model build up plotted against the number of scans. Although the minimum distance and minimum and maximum distance methods appear to perform better, the hybrid method also performs well in terms of surface coverage and outperform the two methods in terms of total number of scans, computational time and termination.

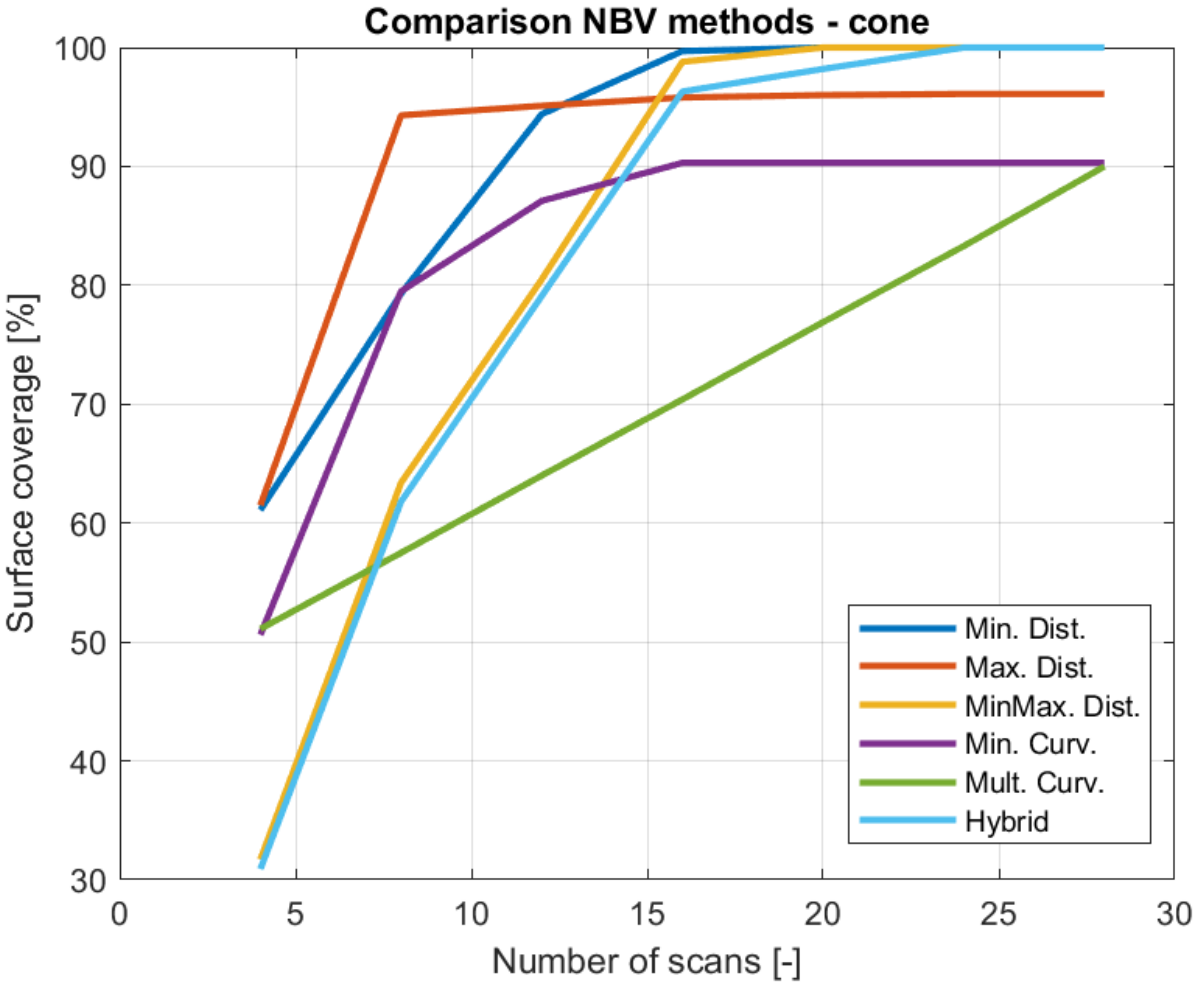


Figure 6.16: *NBV comparison - cone*

The minimum distance method covered the entire object and has nearly symmetrical NBV distribution. This is depicted from two different angles in Figure 6.17 and Figure 6.18.

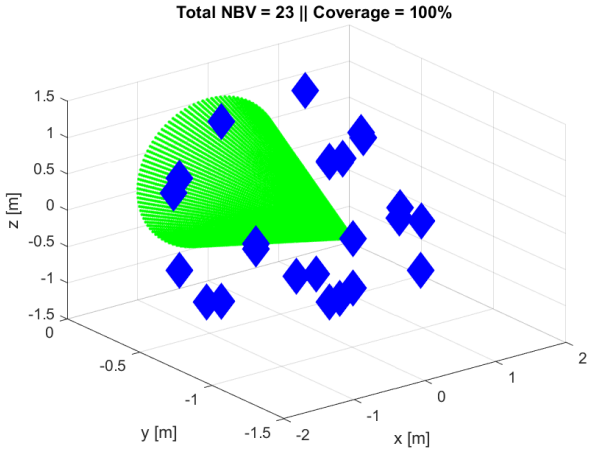


Figure 6.17: *Simulation result - minimum distance - cone - angled view*

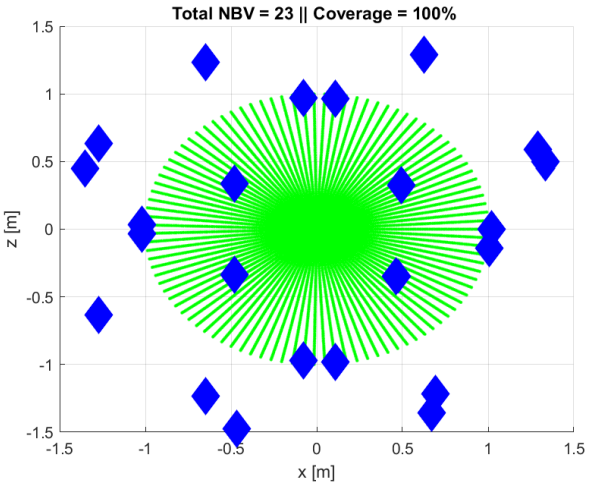


Figure 6.18: *Simulation result - minimum distance - cone - front view*

The maximum distance method also for the cone dominantly moves in the z-direction and remains there, leaving the last parts of the x-direction unscanned. This is illustrated in the Figures 6.19 and 6.20). There is also a unscanned area in the upper part of the cone.

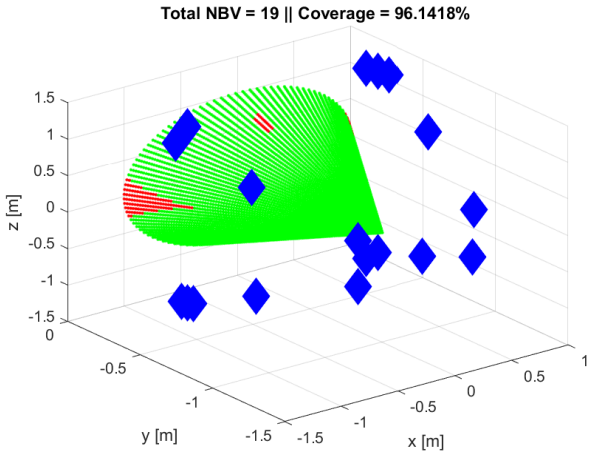


Figure 6.19: *Simulation result - maximum distance - cone - angled view*

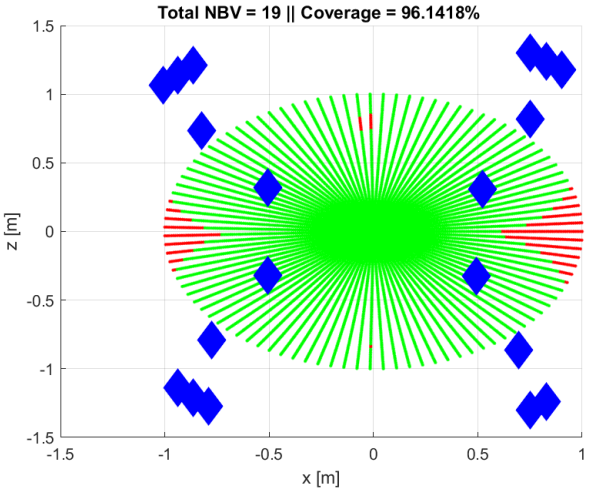


Figure 6.20: *Simulation result - maximum distance - cone - front view*

The minimum and maximum distance method covers the entire cone. There are NBV positions which are near to each other (Figures 6.21 and 6.22).

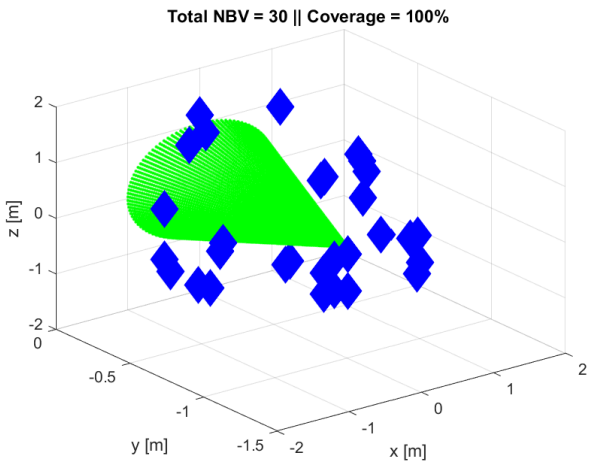


Figure 6.21: *Simulation result - minimum and maximum distance - cone - angled view*

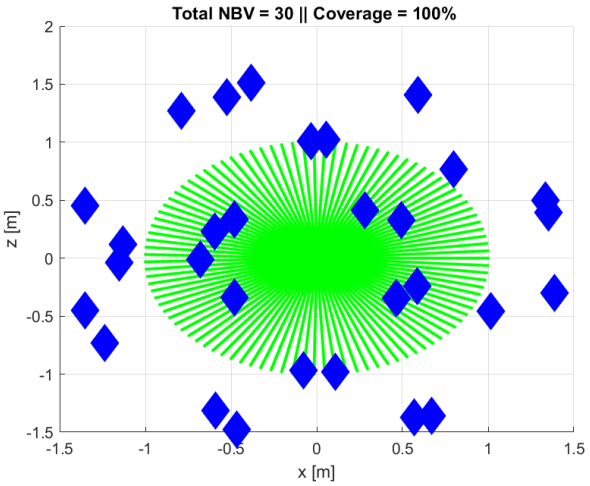


Figure 6.22: *Simulation result - minimum and maximum distance - cone - front view*

Also for the cone, the minimum curvature method also dominantly moves in the z-direction. Therefore the x-axis is not completely visited and remains partially unscanned. Final NBV locations are close to each other (Figures 6.23 and 6.24). The assymetry in the distribution of the NBVs is can be attributed to the fact that there is also asymmetry in the loaded point cloud data. This is visualized in Figure 6.25, so there is not an equal division of the amount of points during partitioning.

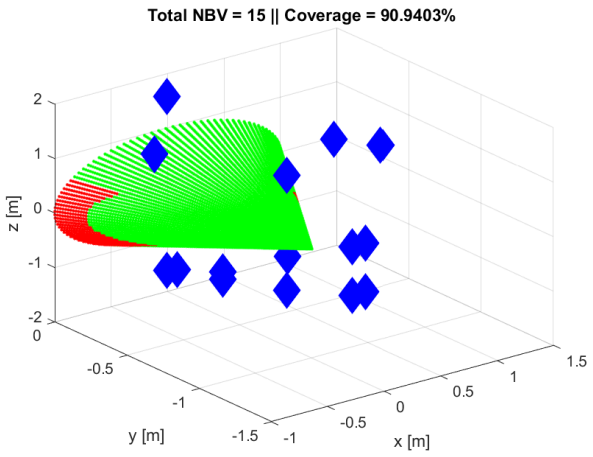


Figure 6.23: *Simulation result - minimum curvature - cone - angled view*

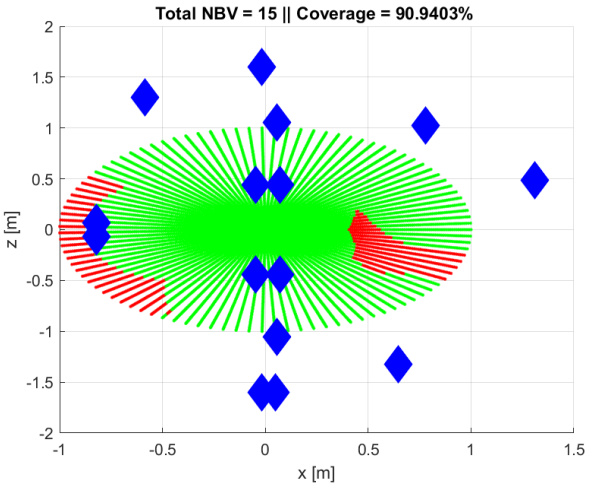


Figure 6.24: *Simulation result - minimum curvature - cone - front view*

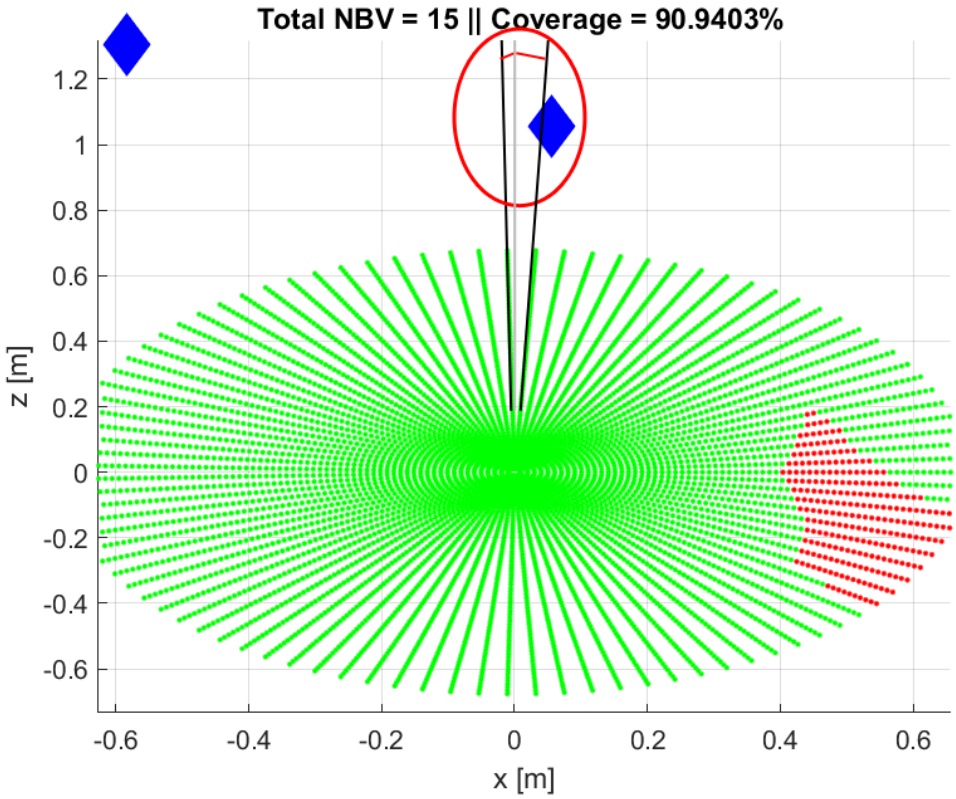


Figure 6.25: *Asymmetric point cloud distribution - cone*

The multiple curvatures method produces leaves the sides unscanned and produces NBvs which are undesirable close to each other. It requires extensive tuning as the curvature values are very small. (Figures 6.26 and 6.27).

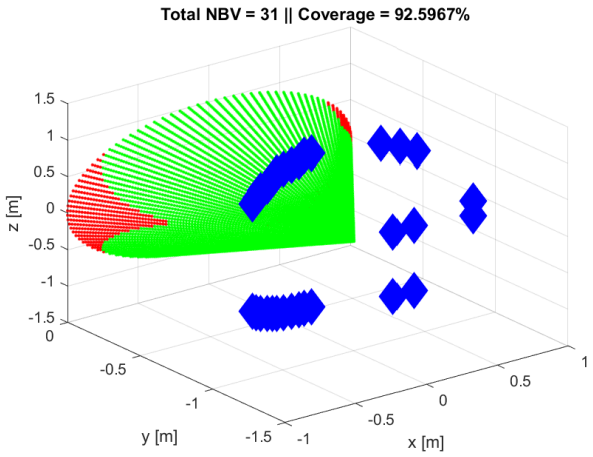


Figure 6.26: *Simulation result - multiple curvatures - cone - angled view*

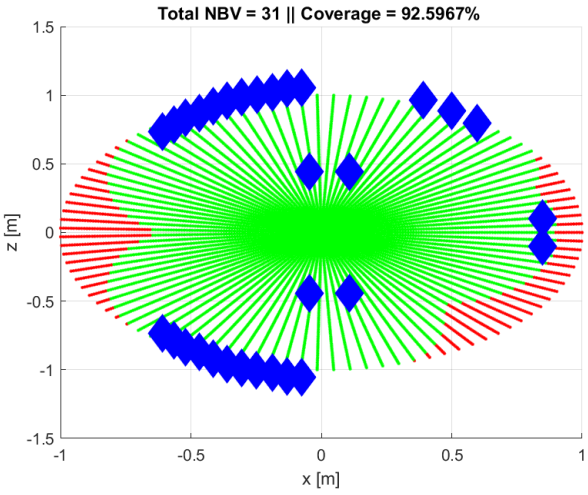


Figure 6.27: *Simulation result - multiple curvatures - cone - front view*

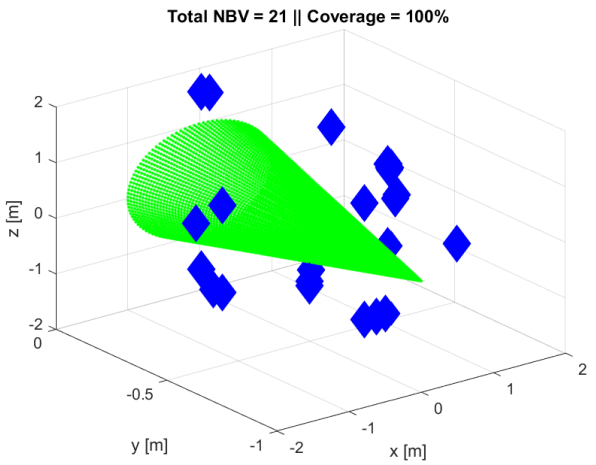


Figure 6.28: *Simulation result - hybrid - cone - angled view*

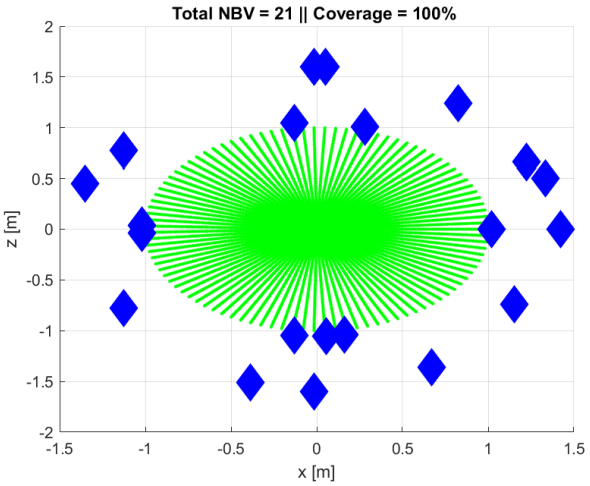


Figure 6.29: *Simulation result - hybrid - cone - front view*

The hybrid method also covers the entire cone. Except for the final scan round, the NBVs are located at acceptable distances and are not close too each other, pointing out to more stable results (Figures 6.28 and 6.29).

The NBV coordinates, the angles ϕ and θ and the scan order from the hybrid method for the cone are tabulated in Table 6.4.

Table 6.4: *NBV results - hybrid method - cone*

Scan [-]	**x** [m]	**y** [m]	**z** [m]	ϕ [°]	θ [°]
1	1.0202	-0.8283	0.0000	45.0000	0
2	0.2796	-0.7923	1.0110	16.4503	50.2417
3	-1.0195	-0.8285	0.0347	-44.9567	1.5048
4	-0.1298	-0.7836	1.0480	-7.5919	52.8314
5	-1.0195	-0.8285	-0.0347	-44.9567	-1.5048
6	-0.1298	-0.7836	-1.0480	-7.5919	-52.8314
7	0.0556	-0.7798	-1.0576	4.5335	-53.5277
8	0.1605	-0.7849	-1.0427	9.5281	-52.4503
9	0.8266	-0.3236	1.2418	31.2123	50.9091
10	1.2235	-0.4500	0.6656	42.5074	25.4161
11	-1.1280	-0.4689	0.7792	-41.1629	30.7774
12	-0.0163	-0.0095	-1.5998	-2.7256	-89.0909
13	-1.1280	-0.4689	-0.7792	-41.1629	-30.7774
14	1.1521	-0.4731	-0.7375	41.6462	-28.9685
15	0.0498	-0.0285	1.5982	4.5335	87.2727
16	1.3362	-0.4194	0.4993	42.6362	18.1818
17	-0.0163	-0.0095	1.5998	-2.7256	89.0909
18	-1.3515	-0.4216	0.4508	-42.9181	16.3636
19	-0.3864	-0.1871	-1.5120	-17.5866	-70.9091
20	1.4243	-0.4243	-0.0000	45.0000	-0.0000
21	0.6712	-0.2817	-1.3596	27.0641	-58.1818

6.3. NBV results for a cubical object

Simulating the cube and the plate resulted in a limitation of the simulation. The pdist2-function of MATLAB activates also the the points of the other sides after the corners, which is not desired. The absence of the normal vector determination, which requires a more detailed new point cloud algorithm which detects corners, and the absence of curvature values, resulted in the use of the maximum and minimum x- and z-values for

NBV selection. The cube and plate needs more research as they differ a lot from a sphere and cube. As also triangulation of a cube and plate is not possible with the current triangulation code, the border points are selected manually.

Figures 6.30 and 6.31 show the simulation result for the cube. Coverage has been omitted as the points at the left, upper, right and lower points are also activated and therefore not add valuable information. It is also possible to cover the four outmost corners of the front side by adding the boundary point which lay in the middle between the maximun x- and z-value.

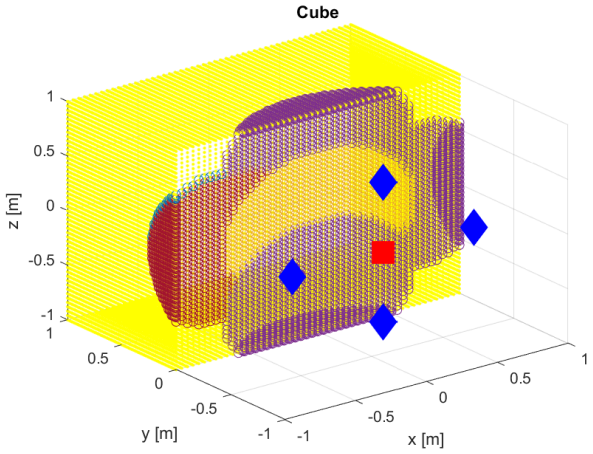


Figure 6.30: *Simulation result of a cube (angled view)*

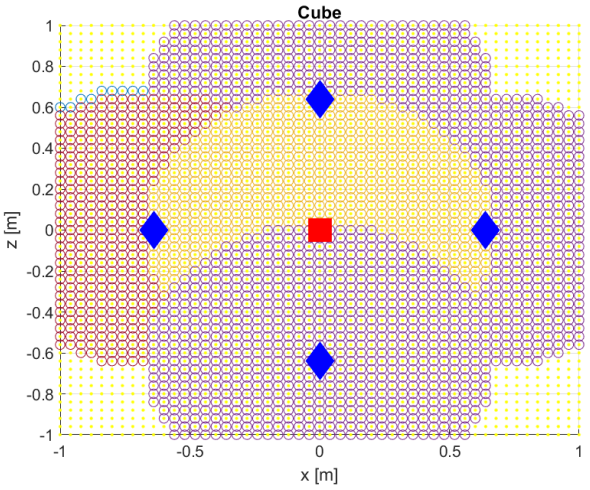


Figure 6.31: *Simulation result of a cube (front view view)*

For the plate, the same conditions hold (Figures 6.32 and 6.33). The plate is largely scanned in four scans. The outer corners can be activated manually and add up another four scans. The NBV determination methods stays the same: selecting the minimum and maximum x- and z-values after the initial scan.

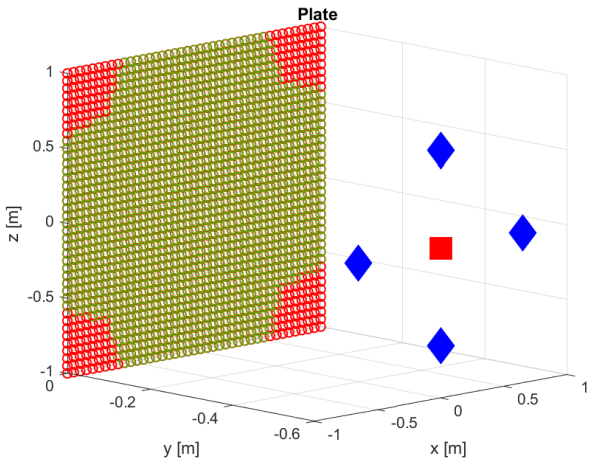


Figure 6.32: *Simulation result of a plate (angled view)*

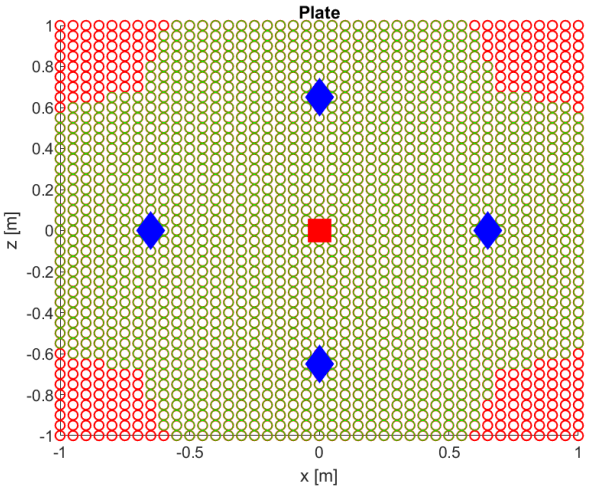


Figure 6.33: *Simulation result of a plate (front view view)*

6.4. NBV results for a spherical object in constrained case

As the LES has several constraints, the following extra simulation has been performed to compare the NBV methods. The ρ is set to 0.58 m and the sensing depth to 0.09 m. This realizes the real LES scan height of 0.4 m in the z-direction (Figure 6.34 and Figure 6.35 - NBV method: minimum distance). The scan width is also 0.4 m instead of the 0.6 m in the real case. This is due to circular working principle of pdist2. Also the maximum ϕ is set under 45 °in the left and right direction. Only the front part of the sphere is scanned, hence the name leading edge in LES. The number of points is, from 5.150 points, brought back to 1.059 points. The

simulation parameters are given in Table 6.5.

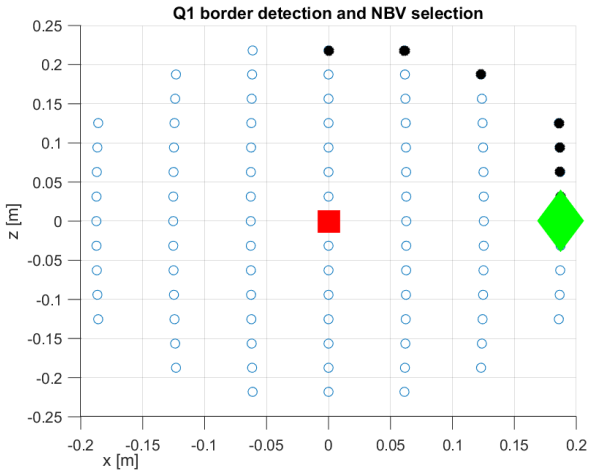


Figure 6.34: *Constrained simulation - NBV selection*

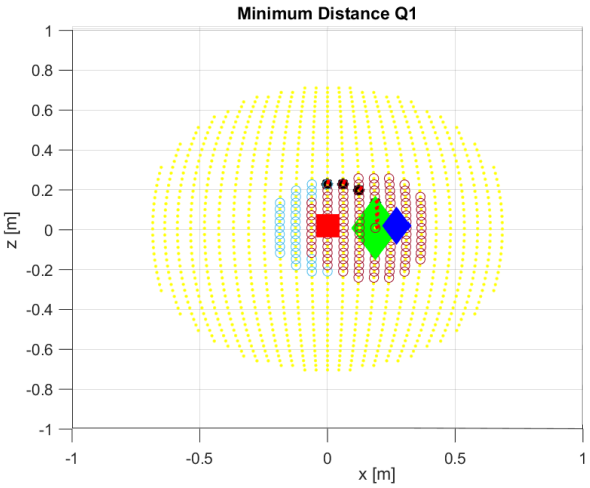


Figure 6.35: *Constrained simulation - NBV selection (detailed)*

Table 6.5: *Simulation parameters - constrained*

Parameter	Value
ρ	0.58
$sense$	0.09
ngb	10
POV	[0, -1.6,0]
Q (sphere)	0 or 5
$dmax$	1
$dista$ (sphere)	0.06 $[m]$

The result from the first scan round after the initial scan of the NBV method minimum distance for the sphere is illustrated in Figure 6.36.

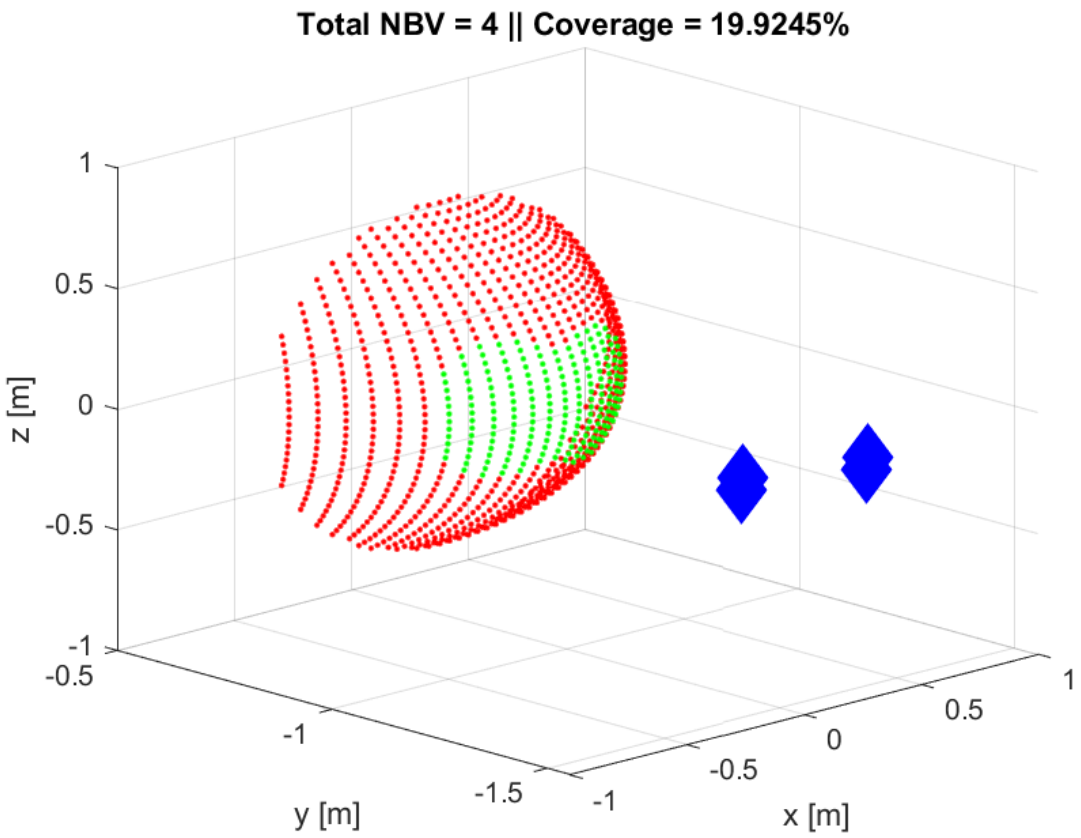


Figure 6.36: *First scan round - sphere - constrained*

The simulation result for the constrained case also resulted in different performances (Table 6.6). None of the NBV methods produce full surface coverage, however the minimum and maximum distance method (99.1 percent) and the tuned hybrid method (99.7 percent) come close. All the methods are terminating well (in 2 or 3 scan rounds). The model build-up is illustrated in Figure 6.37. The minimum and maximum distance method performs well, however the tuned hybrid methods has slight more surface coverage at the end.

Table 6.6: *Constrained NBV results for the sphere*

Method	**Scans [-]**	**Coverage [%]**	**Time [s]**	**Total distance [m]**	**Termination round**
Minimum Distance	20	87.7	13.1	12.2	3
Maximum Distance	18	69.4	18.2	18.7	3
Min+Max Distance	32	99.1	16.1	25.2	2
Minimum Curvature	14	78.9	15.9	11.2	2
Minimum Curvatures	34	83.9	11.5	20.5	2
Hybrid	23	98.4	15.2	19.5	2
Hybrid (tuned, Q=2)	24	99.7	16	23	2

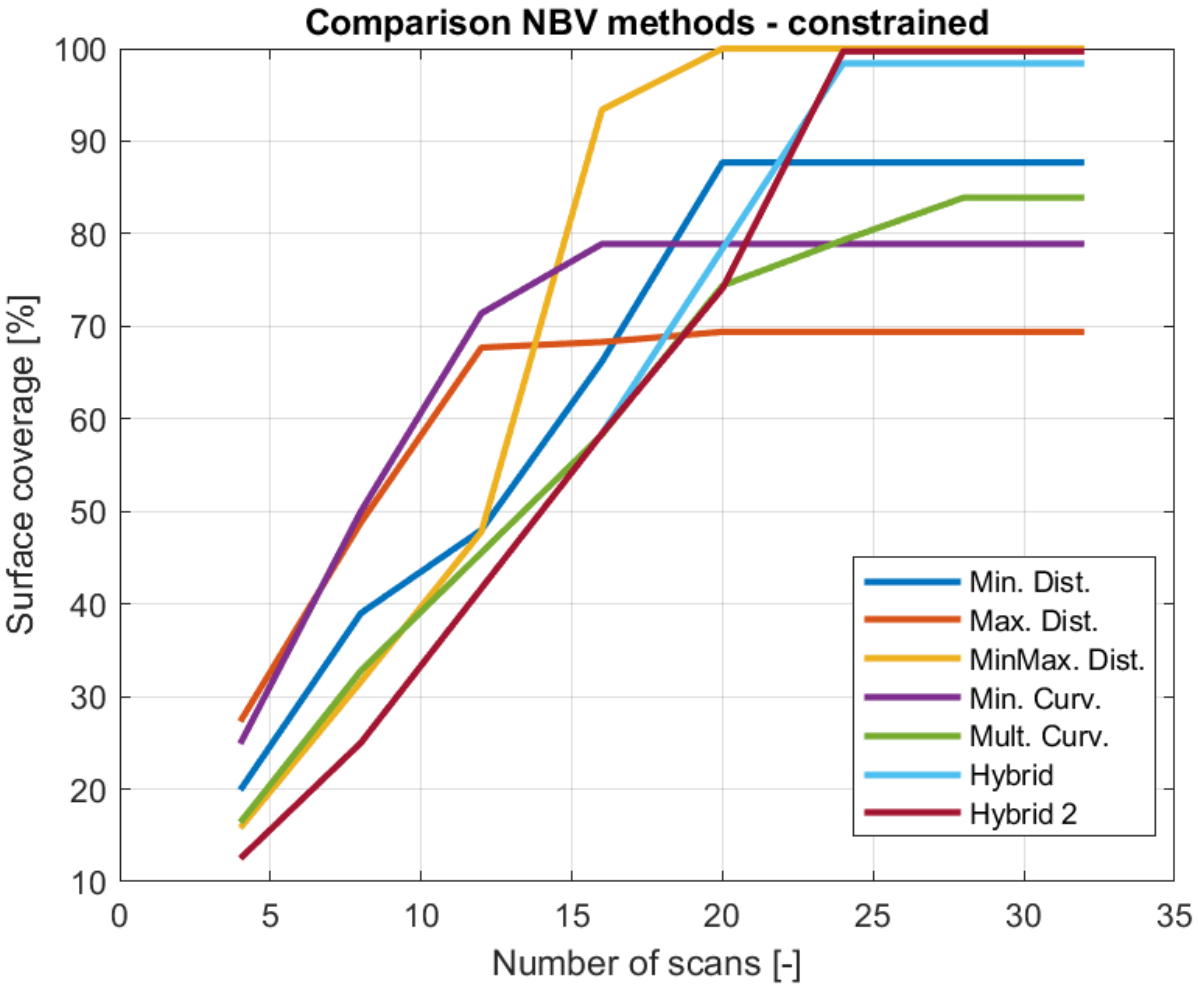


Figure 6.37: *NBV comparison - constrained*

The minimum distance method for NBV selection from constrained scanning leaves the outer portions of the object unscanned. This is depicted from two different angles in Figure 6.38 and Figure 6.39.

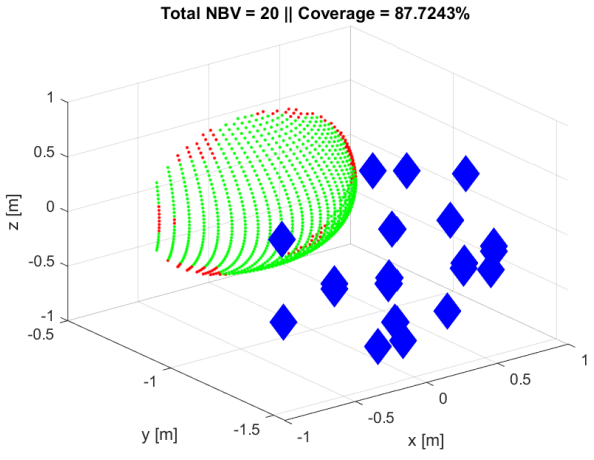


Figure 6.38: *Simulation result - minimum distance - constrained - angled view*

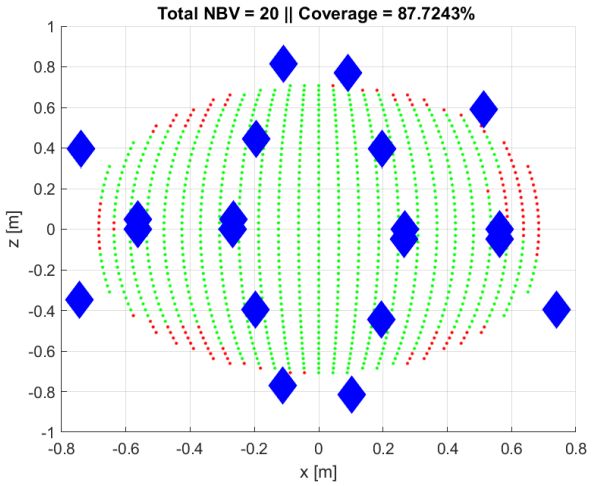


Figure 6.39: *Simulation result - minimum distance - constrained - front view*

The maximum distance method dominantly moves in the diagonals and remains there, leaving large parts of

the object unscanned, hence the low surface coverage of 69.4 percent. This is illustrated in the the Figures 6.40 and 6.41.

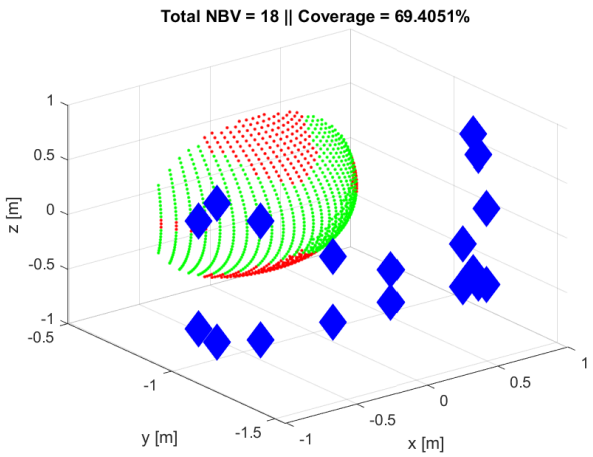


Figure 6.40: *Simulation result - maximum distance - constrained - angled view*

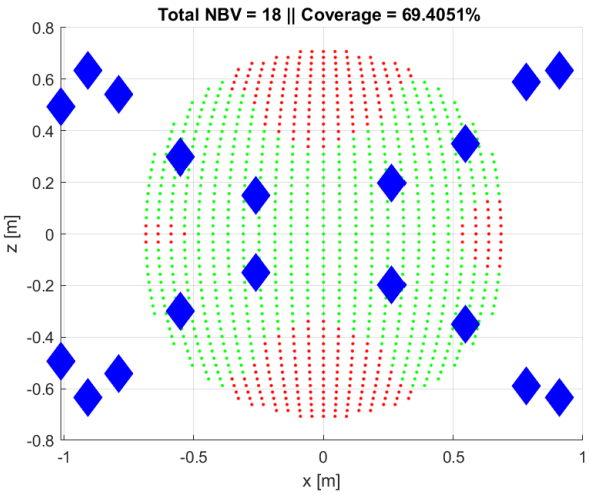


Figure 6.41: *Simulation result - maximum distance - constrained - front view*

The minimum and maximum distance method covers the entire cone. The NBV distribution is symmetrical. Only one percent remains unscanned. (Figures 6.42 and 6.43).

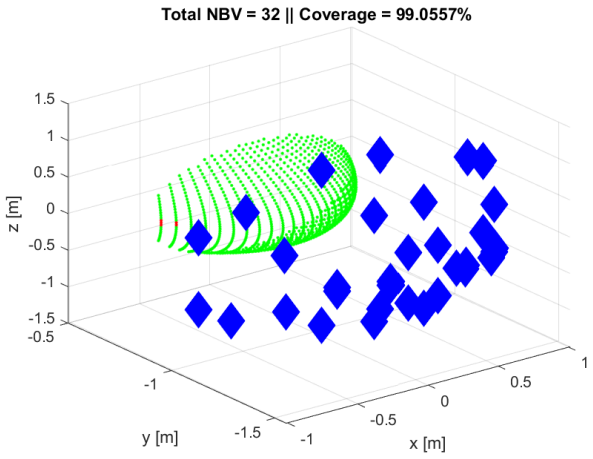


Figure 6.42: *Simulation result - minimum and maximum distance - constrained - angled view*

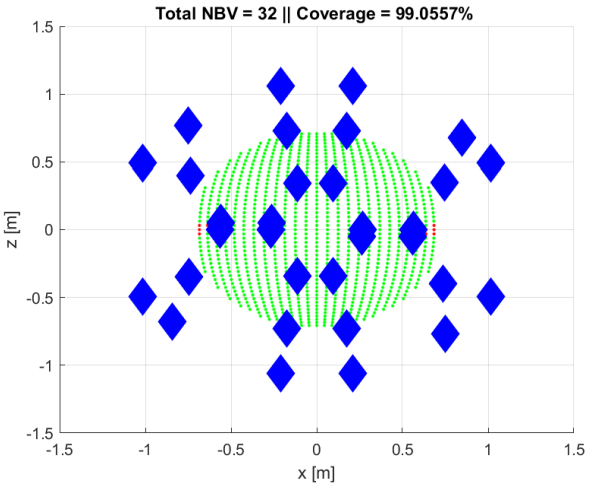


Figure 6.43: *Simulation result - minimum and maximum distance - constrained - front view*

The minimum curvature method moves in z-direction and to the left. The right part remains unscanned. Final NBV locations are close to each other (Figures 6.44 and 6.45).

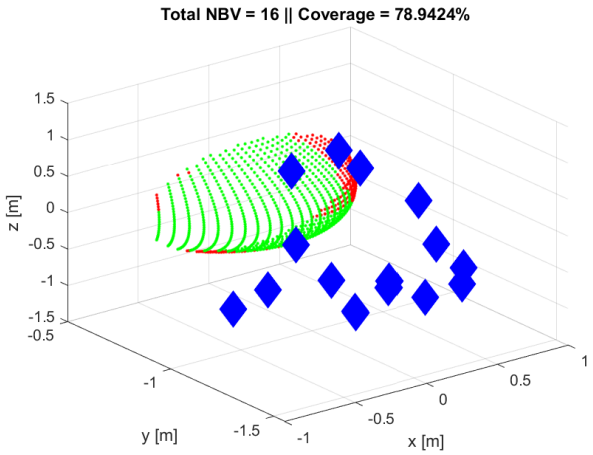


Figure 6.44: *Simulation result - minimum curvature - constrained - angled view*

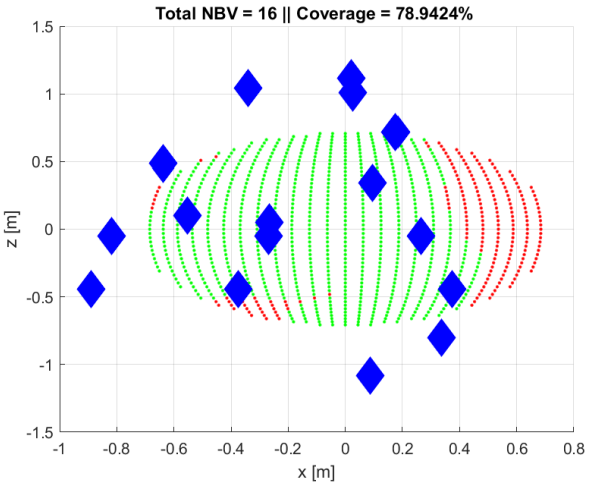


Figure 6.45: *Simulation result - minimum curvature - constrained - front view*

The multiple curvatures method produces a lot on NBVs with low information gain. Major parts remain unscanned. (Figures 6.46 and 6.47).

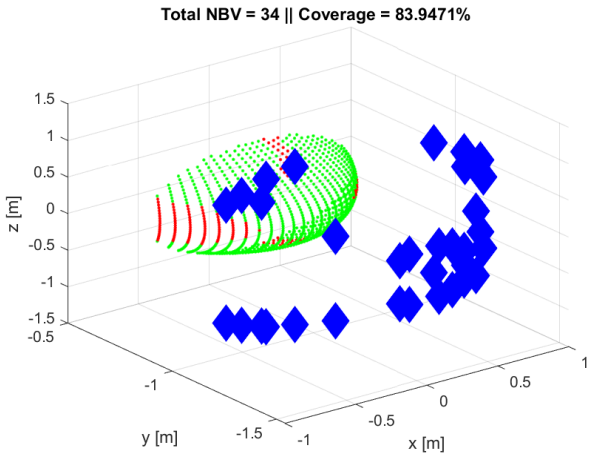


Figure 6.46: *Simulation result - multiple curvatures - constrained - angled view*

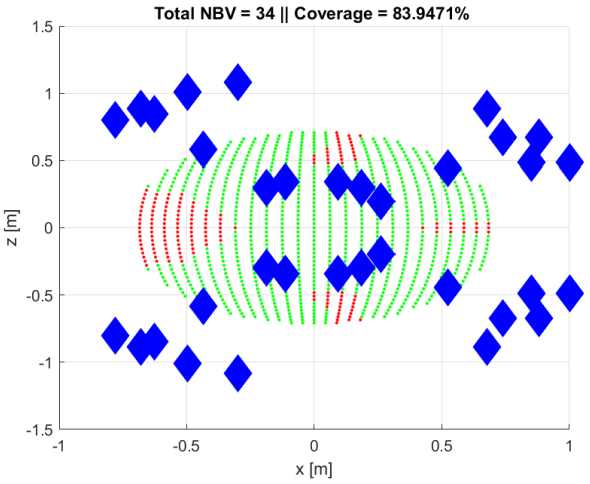


Figure 6.47: *Simulation result - multiple curvatures - constrained - front view*

The hybrid method misses 1.6 percent of the coverage (Figures 6.48 and 6.49). The benefit of the hybrid method is that it is tunable, which is done in the next method.

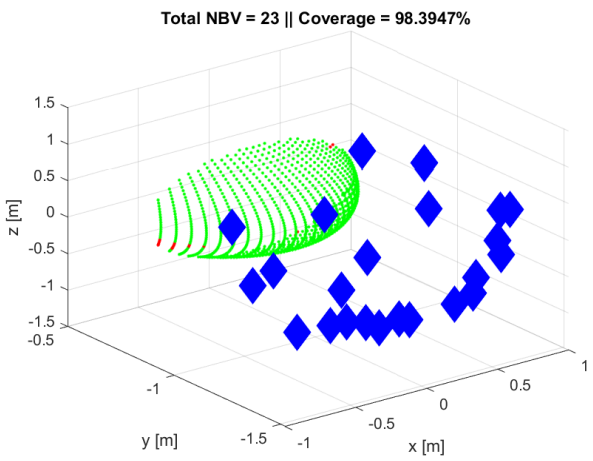


Figure 6.48: *Simulation result - hybrid - constrained - angled view*

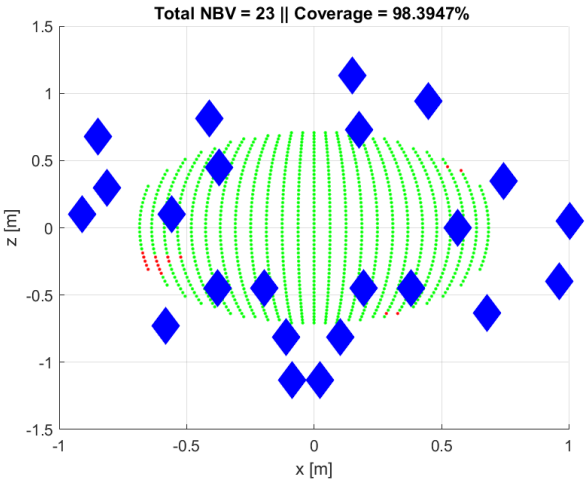


Figure 6.49: *Simulation result - hybrid - constrained - front view*

The tuned hybrid method results in the most surface coverage (Figures 6.50 and 6.51. Mainly all the NBVs are distributed evenly. There is more optimization possible.

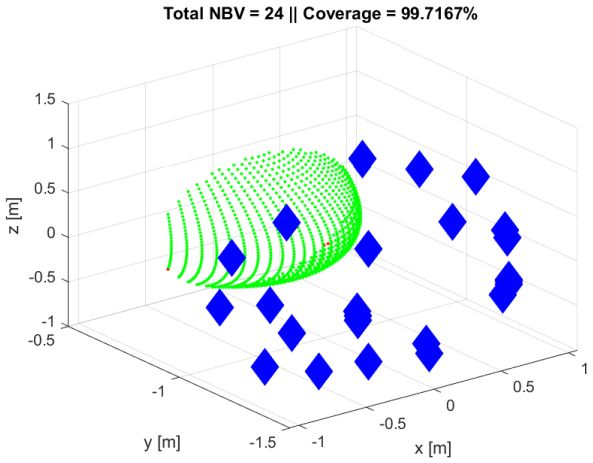


Figure 6.50: *Simulation result - hybrid 2 - constrained - angled view*

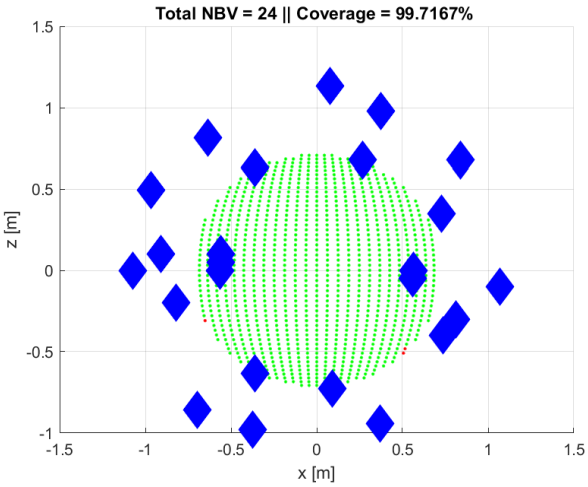


Figure 6.51: *Simulation result - hybrid 2 - constrained - front view*

The NBV coordinates, the angles ϕ and θ and the scan sequence from the hybrid method for the sphere are tabulated in Table 6.7. The scanning takes place in rounds and NBV positions are selected anti-clockwise (from Q1 to Q2, Q2 to Q3 and Q3 to Q4).

Table 6.7: *NBV coordinates and angles - hybrid method - constrained*

Scan [-]	**x** [m]	**y** [m]	**z** [m]	ϕ [°]	θ [°]
1	0.2656	-1.4231	0.6812	10.1951	25.2000
2	0.5620	-1.4976	0.0000	18.8538	0
3	-0.5597	-1.4950	0.1005	-18.7319	3.6000
4	-0.5617	-1.4968	0.0503	-18.8543	1.8000
5	-0.3595	-1.4237	-0.6354	-13.7939	-23.4000
6	-0.5620	-1.4976	0.0000	-18.8538	0
7	0.0913	-1.4227	-0.7264	3.7913	-27.0000
8	0.5617	-1.4968	-0.0503	18.8543	-1.8000
9	0.3749	-1.2072	0.9807	16.0014	37.8000
10	0.7287	-1.3806	0.3490	26.2027	12.6000
11	-0.9079	-1.3127	0.1005	-32.4522	3.6000
12	-0.3595	-1.4237	0.6354	-13.7939	23.4000
13	-0.3740	-1.2075	-0.9807	-15.8888	-37.8000
14	-0.8205	-1.3581	-0.2005	-29.0358	-7.2000
15	0.8141	-1.3437	-0.2998	29.2304	-10.8000
16	0.7387	-1.3623	-0.3979	27.9138	-14.4000
17	0.0793	-1.1285	1.1314	4.7150	45.0000
18	0.8375	-1.1807	0.6812	34.2672	25.2000
19	-0.9680	-1.1741	0.4944	-39.3505	18.0000
20	-0.6367	-1.2206	0.8145	-25.4609	30.6000
21	-0.6955	-1.1574	-0.8573	-28.6718	-32.4000
22	-1.0738	-1.1855	0.0000	-40.4538	0
23	1.0708	-1.1840	-0.1005	40.3319	-3.6000
24	0.3703	-1.2397	-0.9405	14.3457	-36.0000

6.5. Optimal NBV algorithm for LES

This Chapter presented the results of the NBV selection methods. The hybrid method results in the better numbers at scanning a sphere and cone in an unconstrained and constrained case. The hybrid methods requires less scans and therefore causes less robot movement, requires less computational time and terminates more stable. The additional benefit of the hybrid method is that it is tunable thanks to the Q, which activates more detected curvature values. This will form a basis for scanning complex and large objects. The main NBV strategy is based on the SEE where border points are detected and normal vectors are calculated.

7
Future Work

The next step after selection of the most optimum NBV strategy and ranking method is the implementation of it in real life. This strategy needs to be coded to the software environment (Beckhoff) of LES to be able to automatically scan objects.

As the current LES setup does not have collision avoidance, the chosen algorithm and method should be extended with an extensive path planning strategy to avoid collision at any cost to prevent damage to both sides. The robot arm does not know that there is an object in the way when it, for example, needs to move from its rightmost point to the left side.

In order to scan complex objects, or other aircraft parts, with self-occlusion, the SEE algorithm could be extended with occlusion detection methods and an extra view adjustment step. This will also help to be able to calculate the right NBVs in case of discontinued surfaces. The additional benefit of SEE with the hybrid NBV selection method is that it can be tuned further. This will form a basis for scanning complex and large objects.

If the scanning and moving time becomes less dominant, scan quality will be even more of importance and if more computation power is added, the point cloud can be voxelized at every step to see if it results in better NBV predictions for defect detection.

The LES can, as its name says, only scan a portion of the front part of an object. The robotic arm is in the current setting, due to its relative short arms, not able to make azimuth angles larger than 45 degrees (to the right and left) to ensure the working distance of 0.6 m. This decreases at larger angles, lowering the quality of the scan or even detoriates it. A new study on the feasibility of adding joints and arms can be made to always ensure the wanted working distance at larger azimuth angles.

8 Conclusion

This thesis describes the concept of view planning based on algorithms for 3D reconstruction of unknown objects. The LES of NLR, a five DOF robotic arm equipped with a monochrome 3D depth sensor and a RGB sensor, is able to scan objects and reconstruct 3D models for maintenance purposes. A 3D model requires several images from different viewpoints. However, currently the selection of the next viewpoints is done manually, especially the positioning of the robot arm and thereby also the sensor pose. It is expected that the LES automatically positions the sensors to the next views to further automate the image acquisition process. Further automating the LES will lead to the final goal of automatically detection an classification of anomalies.

In order to reconstruct a model in 3D, knowledge about the required hardware and data acquisition is needed. Robots and sensors are used to automatically build a model in 3D. Obtained data about the environment is mainly based on three representations: voxels, triangle meshes and point clouds.

To achieve the goal of automated positioning of the LES to new locations in its workspace in order to fully reconstruct a 3D model, an algorithm is needed. Therefore, relevant NBV approaches and selection methods in literature, which should be applicable to the LES, are compared in order to find the most suitable algorithm for the LES.

Finding the NBV remains a challenging problem, which has been being studied since the 1980s. NBV algorithms are divided into two groups: model based or non-model based. Latter does not have a priori information about the object nor its geometry is known. Algorithms are also further divided in the two classifications search-based and volumetric. Search-based approaches firstly produce a large numbers of candidate viewpoints which are then filtered or selected under defined constraints. It makes use of measurements. Volumetric methods use the workspace as a basis for analysis and is mainly about the so called voxelization of occupied and empty areas. The computer memory consumption is high. Surface methods aim at surfaces of objects where edges are sought-after to compute occluded area. Basic volumetric NBV algorithms are inaccurate and unreliable in case of complex occlusion and uncertainty about positions.

The obtained knowledge about NBV algorithms and results of relevant simulations and experiments lead to the research question **what the optimal NBV algorithm for 3D reconstruction of unknown objects with the five DOF LES** of NLR will be. This research question is answered with the aid of answers to the following three sub-questions: (1) 'How is the imaging pipeline set up for image acquisition and object registration?', (2) 'What sensor systems are required for unknown object reconstruction?' and (3) 'What are the prominent view planning algorithms?' Also, answers on all above questions supported the reasoning behind the choice of the NBV algorithm, which is explained below.

A 3D model can be acquired by point clouds, which are stitched together after scan rounds. Other data structures are the triangle mesh and voxels. The reconstruction workflow consists of robot positioning, scanning, registration and update and planning the NBV. This process continues until a termination criterion is met.

Automatic image acquisition is done by robots and sensors. The three main sensor types are stereo vision. ToF and structured light. Robots can be robotic arms or independent moving full-size robots. The LES consists of a robotic arm using a 3D depth sensor combined with a 2D RGB camera.

Many NBV algorithms with different constraints exists. Connolly was one of the first to calculate the NBVs via an evenly sampled sphere around the object. Pito further advanced the work of Connolly by considering several constraints, requirements and output quality. Banta et al. used a voxel representation. Chen and Li focused on the surface trend. Vasquez-Gomez et al. and Kriegel et al. researched the IG. The conclusion of Border et al. was that their algorithm called SEE outperformed state-of-the-art volumetric methods.

The applicable algorithm for LES is researched to be the SEE, where the basis is border detection and normal vector calculation, and using the ranking method of minimum distance and minimum curvature combined (hybrid version). It ensures full surface coverage at an acceptable computational time and total sensor movement distance. It also terminates more stable than the other only distance and curvature methods, which most of them not being capable of scanning an entire object.

Bibliography

[1] R. Zeng, Y. Wen, W. Zhao, and Y. Liu. View planning in robot active vision: A survey of systems, algorithms and applications. *Computational Visual Media 6*, pages 225–245, 2020.

[2] W. R. Scott, G. Roth, and J. Rivest. View planning for automated three-dimensional object reconstruction and inspection. *ACM Computing Surveys*, Vol 35(No. 1):64–96, 2003.

[3] S. H. Wu, W. Sun, P. X. Long, H. Huang, D. Cohen-Or, M. L. Gong, O. Deussen, and B. Q. Chen. Quality-driven Poisson-guided autoscanning. *ACM Transactions on Graphics*, Vol. 33(No. 6, Article No. 203), 2014.

[4] D. Marr and T. Poggio. A computational theory of human stereo vision. *Proceedings of the Royal Society B: Biological Sciences*, Vol. 204 - 1156:301–328, 1979.

[5] Xing. Lu, Lui D. Zhou, J., and J. Zhang. Application of color structured light pattern to measurement of large out-of-plane deformation. *Acta Mechanica Sinica*, Vol. 27 - 6:1098–1104, 2011.

[6] I. Jovancevic, H. Pham, J. Orteu, R. Gilblas, J. Harvent, X. Maurice, and L. Brethes. 3D Point Cloud Analysis for Detection and Characterization of Defects on Airplane Exterior Surface. *Nondestructive Evaluation*, 36, 2017.

[7] C. McGreavy, L. Kunze, and N. Hawes. Next Best View Planning for Object Recognition in Mobile Robotics. *Proceedings of the 34th Workshop of the UK Planning and Scheduling Special Interest Group (PlanSIG)*, Vol. 1782, 2016.

[8] S. Chen, Y. Li, J. Zhang, and W. Wang. *Active sensor planning for multiview vision tasks*, volume 1. Berlin Heidelberg: Springer, 2008.

[9] J. I. Vasquez-Gomez, L. E. Sucar, R. Murrieta-Cid, and E. Lopez-Damian. Volumetric next-best-view planning for 3D object reconstruction with positioning error. *International Journal of Advanced Robotic Systems*, 2014.

[10] G. H. Tarbox and S. N. Gottschlich. Planning for complete sensor coverage in inspection. *Computer Vision and Image Understanding*, Vol. 61(No. 1):84–111, 1995.

[11] M. Mendoza, J. I. Vasquez-Gomez, H. Taud, L. E. Sucar, and C. Reta. Supervised learning of the next-best-view for 3D object reconstruction. *Pattern Recognition Letters*, Vol. 133:224–231, 2020.

[12] J. Delmerico, S. Isler, R. Sabzevari, and D. Scaramuzza. A comparison of volumetric information gain metrics for active 3D object reconstruction. *Autonomous Robots*, Vol. 42(No. 2):197–208, 2018.

[13] A. Doumanoglou, R. Kouskouridas, S. Malassiotis, and T. K. Kim. Recovering 6D object pose and predicting next-best-view in the crowd. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, page 3583–3592, 2016.

[14] N. A. Massios and R. B. Fisher. A best next view selection algorithm incorporating a quality criterion. *Proceedings of the British Machine Vision Conference*, page 780–789, 1998.

[15] M. Krainin, B. Curless, and D Fox. Autonomous generation of complete 3D object models using next best view manipulation planning. *Proceedings of the IEEE International Conference on Robotics and Automation*, page 5031–5037, 2011.

[16] R. Border, J. D. Gammell, and P. Newman. Surface Edge Explorer (SEE): Planning Next Best Views Directly from 3D Observations. *IEEE International Conference on Robotics and Automation (ICRA)*, 2018.

[17] R. Pito. A solution to the next best view problem for automated surface acquisition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol 21(No. 10), 1999.

[18] R. Border and J. D. Gammell. Proactive estimation of occlusions and scene coverage for planning next best views in an unstructured representation. *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020.

[19] J. I. Vasquez-Gomez, L. E. Sucar, and R. Murrieta-Cid. View planning for 3D object reconstruction with a mobile manipulator robot. *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2014.

[20] Meagher D. Geometric modeling using octree encoding. *Computer Graphics and Image Processing*, Vol. 19(No. 2):129–147, 1982.

[21] J. I. Vasquez-Gomez, L. E. Sucar, and R. Murrieta-Cid. Hierarchical ray tracing for fast volumetric next-best-view planning. *International Conference on Computer and Robot Visions*, 2013.

[22] C. Connolly. The determination of next best views. *Proceedings of the IEEE International Conference on Robotics and Automation*, page 432–435, 1985.

[23] J. Maver and R. Bajcsy. Occlusions as a guide for planning the next view. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol 15(No. 5), 2013.

[24] X. Yuan. A mechanism of automatic 3D object modelling. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 307–311, 1995.

[25] D. Papadopoulos-Orfanos and Schmitt F. Automatic 3-D digitization using a laser rangefinder with a small field of view. *Proceedings of the International Conference on Recent Advances in 3-D Digital Imaging and Modeling*, 1997.

[26] M. Reed and P. Allen. Constraint-based sensor planning for scene modelling. *IEEE Transactions Pattern Analysis and Machine Intelligence*, pages 1460–1467, 2000.

[27] J Banta, L. Wong, and Abidi M. A next-best-view system for autonomous 3-D object reconstruction. *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, 2000.

[28] M Callieri, A. Fasano, G. Impoco, P. Cignoni, R. Scopigno, Parrini G., and Biagini G. RoboScan: an automatic system for accurate and unattended 3D scanning. *3D Data Processing, Visualization and Transmission (3DPVT)*, 2004.

[29] S. Chen and Y. Li. Vision sensor planning for 3-D model acquisition. *IEEE Transactions on Systems, Man and Cybernetics*, pages 894–904, 2005.

[30] S. Kriegel, T. Bodenmüller, M. Suppa, and Hirzinger G. A surface-based next-best-view approach for automated 3D model completion of unknown objects. *IEEE International Conference on Robotics and Automation*, 2011.

[31] M. Karaszewski, R. Sitnik, and E. Bunsch. On-line, collision-free positioning of a scanner during fully automated three-dimensional measurement of cultural heritage objects. *Robotics and Autonomous Systems*, pages 1205–1219, 2012.

[32] S. Khalfaoui, R. Seulin, Y. Fougerolle, and D. Fofi. An efficient method for fully automatic 3D digitization of unknown objects. *Computers in Industry*, Vol. 64(No. 9):1152–1160, 2013.

[33] A. H. Ahmadabadian, S. Robson, J. Boehm, and M. Shortis. Stereo-imaging network design for precise and dense 3d reconstruction. *The Photogrammetric Record 29(147)*, pages 317–336, 2014.

[34] J. E. Bresenham. Algorithm for computer control of a digital plotter. *IBM Systems Journal*, page 4:25–30, 1965.

[35] K. M. Wurm, A. Hornung, M. Bennewitz, C. Stachniss, and W. Burgard. OctoMap: A probabilistic, flexible, and compact 3D map representation for robotic systems. *Proc. of the ICRA 2010 Workshop on Best Practice in 3D Perception and Modeling for Mobile Manipulation*, 2010.

[36] S. Kriegel, C. Rink, T. Bodenmüller, and M. Suppa. Efficient next-best-scan planning for autonomous 3D surface reconstruction of unknown objects. *Journal of Real-Time Image Processing volume*, Vol. 10:611–631, 2013.

[37] M. Karaszewski, M. Adamczyk, and R. Sitnik. Assessment of next-best-view algorithms performance with various 3D scanners and manipulator. *ISPRS Journal of Photogrammetry and Remote Sensing*, Vol. 119:320–333, 2016.

[38] M. Awrangjeb. Boundary extraction (identification and tracing) from point cloud data - MATLAB Central File Exchange. https://www.mathworks.com/matlabcentral/fileexchange/60690-boundary-extraction-identification-and-tracing-from-point-cloud-data. July 2021.

[39] R. B. Rusu. Semantic 3D object Maps for Everyday Manipulation in Human Living Environments. *PhD thesis, Technische Universität München*, pages 37–50, 2010.

[40] M. Awrangjeb. Using point cloud data to identify, trace and regularize the outlines of buildings. *International Journal of Remote Sensing*, Vol. 37:551–579, 2016.

[41] Z. Taylor. Find 3D Normals and Curvature - MATLAB Central File Exchange. https://www.mathworks.com/matlabcentral/fileexchange/48111-find-3d-normals-and-curvature. May 2021.

A
Appendix A

This Appendix presents the main MATLAB code where the outline can be found in Section 5.1.2. Only the calculations of Q1 are shown as Q2, Q3 and Q4 are using the same principle.

```
clear all; clc; close all

tic

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%                  CHOOSE INPUT: 1) Cone or 2) Sphere                        %%

 coor = load('sphere_100.txt');        % Uncomment this for the SPHERE (Unconstrained)

% coor = load('cone.txt');             % Uncomment this for the CONE (Unconstrained)

% coor = load('sphere_front.txt');     % Uncomment this for the SPHERE (constrained)


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%                                  PARAMETERS                                %%


rho = 0.6;             % Working distance of sensor [Unconstrained]
sense = 0.3;           % Sensing depth [Unconstrained]

% rho = 0.58;          % Working distance of sensor [Constrained]
% sense = 0.09;        % Sensing depth [Constrained]

Q = 0.0;               % Tune value for curvature(s)
                       % [0= one minimum curvature value, higher number for more NBVs]
% Q = 5.5e11;          % Q voor cone

dista = 0.1;           % SPHERE: Distance between NBVs [m] (for optimization),
                       % set 0 for activating all points
% dista = 0.01597;       % CONE
% dista = 0.06;          % UNCONSTRAINED CASE

neighbors = 10;        % Number of neighbors for local geometry analysis
dmax = 1;              % Max point-to-point distance for boundary detection [1=standard]

boundx = 1;            % Object Boundary Box (OBB) in +x direction
boundz = 1;            % Object Boundary Box (OBB) in +z direction

boundx_constrained = 0.8; % CONSTRAINED: Object Boundary Box (OBB) in +x direction
boundz_constrained = 0.8; % CONSTRAINED: Object Boundary Box (OBB) in +z direction

POV = [0 -1.6 0];      % Initial scan location, default = [0 -1.6 0]
POV_Q1 = [1 -1.6 1];  % Normal vector sensor direction Q1, default = [1 -1.6 1]
POV_Q2 = [-1 -1.6 1]; % Normal vector sensor direction Q2, default = [-1 -1.6 1]
POV_Q3 = [-1 -1.6 -1];% Normal vector sensor direction Q3, default = [-1 -1.6 -1]
POV_Q4 = [1 -1.6 -1]; % Normal vector sensor direction Q4, default = [1 -1.6 -1]

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
coor = coor(~all(coor == 0, 2),:);
coor = unique(coor, 'rows', 'stable');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%                          Matrix initialization                          %%


ALL_coor = [];
ALL_POV_F=[];
ALL_POV_Fold=[];

POVQ1 = [];
POVQ2 = [];
POVQ3 = [];
POVQ4 = [];
coorQ1 = [];
coorQ2 = [];
coorQ3 = [];
coorQ4 = [];

ALL_coor_np  = [];

    result = pdist2(coor(:,:),POV);
    flt = result<=rho+sense ; %% 0.6 m working distance + 0.3 m diepte
    coor2 = coor(flt,:);

x = coor2(:,1); y = coor2(:,2); z = coor2(:,3);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Matrix partitioning in the quadrants Q1, Q2, Q3 and Q      %%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Q1 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Q1 = coor2(:,3)>=0 & coor2(:,1)>=0;
Q1 = Q1.*coor2;
Q1 = Q1(~all(Q1 == 0, 2),:);
xQ1=Q1(:,1); yQ1=Q1(:,2); zQ1=Q1(:,3);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Q2 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Q2 = coor2(:,3)>0 & coor2(:,1)<0;
Q2 = Q2.*coor2;
Q2 = Q2(~all(Q2 == 0, 2),:);
xQ2=Q2(:,1); yQ2=Q2(:,2); zQ2=Q2(:,3);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Q3 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Q3 = coor2(:,3)<=0 & coor2(:,1)<=0;
Q3 = Q3.*coor2;
Q3 = Q3(~all(Q3 == 0, 2),:);
xQ3=Q3(:,1); yQ3=Q3(:,2); zQ3=Q3(:,3);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Q4 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Q4 = coor2(:,3)<0 & coor2(:,1)>0;
Q4 = Q4.*coor2;
Q4 = Q4(~all(Q4 == 0, 2),:);
xQ4=Q4(:,1); yQ4=Q4(:,2); zQ4=Q4(:,3);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Q1 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%%          Normal vector and curvature calculations    %%

[normals,curvature] = findPointNormals(Q1,neighbors,
% [POV_Q1(:,1) POV_Q1(:,2) POV_Q1(:,3)],false); % NORMAL VECTOR DIRECTION!!!
angles_azimuth = atan2(normals(:,2),normals(:,1));
Punt = sqrt(xQ1.^2+yQ1.^2+zQ1.^2);
angle_gamma = acos(zQ1./Punt);  %% ELEVATION!
matrix = [xQ1,yQ1,zQ1,angles_azimuth,angle_gamma
% normals(:,1), normals(:,2),normals(:,3),curvature];

%%           Border point detection via triangulation                  %%

XA = [x,z];
rect1 = [min(x),min(z); max(x),min(z); max(x),max(z); min(x),max(z); min(x),min(z)];
IN1 = inpolygon(XA(:,1), XA(:,2), rect1(:,1), rect1(:,2));
XA1 = XA(IN1,:);

[bids1 E Ne] = find_delaunay_boundary03_fig1(XA,dmax);

if bids1{1}(1,1) == bids1{1}(end,1)
    bids1{1} = bids1{1}(1:end-1,:);
end

bs1 = XA1(bids1{1},:);
bsp = [bs1];
[bids E Ne] = find_delaunay_boundary03_fig1(bsp,dmax);


%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%     Matrix Matching - Border points with coordinates     %%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

coor_swapQ1 = [Q1(:,1),Q1(:,3),Q1(:,2)]; % Change column order from x,y,z to x,z,y

A = coor_swapQ1;
B = bsp;

Border_test = ismember(A(:,1:2),B,'rows');
Border = Border_test.*coor_swapQ1;

Border = Border(~all(Border == 0, 2),:);
Border = unique(Border, 'rows', 'stable');
Border = Border(~all(Border == 0, 2),:);

Border = [Border(:,1),Border(:,3),Border(:,2)];
% Return back to column order x,y,z (instead of x,z,y)

%%

C = Border(:,1:3);
D = matrix(:,1:3);
Border9 = ismember(D,C,'rows');
Border9 = Border9.*matrix(:,:);
Border9 = Border9(~all(Border9 == 0, 2),:);
Border9 = Border9(logical(Border9(:,2)),:);
```

```
figure
set(gcf,'color','w');
scatter3(x,y,z)
hold on
scatter3(Border(:,1), Border(:,2),Border(:,3),'k*','LineWidth',5)
title('Border detection Q1 - initial scan')
xlabel('x [m]')
ylabel('y [m]')
zlabel('z [m]')
view([0 0])

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% MIN CURVATURE Q1

[min_curv,curvature_I] = min(Border9(:,9));
min_curvature = Border9(curvature_I,:);


min_curv_all = min_curv+(min_curv*Q);              % TUNE
Min_curv_test= Border9(:,9)<=min_curv_all;         % TUNE
All_min_curv = Min_curv_test.*Border9(:,:);
All_min_curv = All_min_curv(~all(All_min_curv == 0, 2),:);

%% MIN DISTANCE Q1

Border9_distance = [Border9(:,1),Border9(:,2),Border9(:,3)];
[distances] = pdist2(Border9_distance(:,:),POV);
My_matrix = [Border9,distances];

[distance_min,I_min] = min(My_matrix(:,10));
Matrix_mindistance = My_matrix(I_min,:);

distance_min_all = distance_min;
Min_dist_test= My_matrix(:,10)<=distance_min_all;
All_min_dist = Min_dist_test.*My_matrix(:,:);
All_min_dist = All_min_dist(~all(All_min_dist == 0, 2),:);
All_min_dist = All_min_dist(round(end/2),:);

All_hybrid = [All_min_curv;All_min_dist(:,1:9)];

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
min_curvature = All_hybrid;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

figure
set(gcf,'color','w');
plot(distances(:,1),'LineWidth',2)
grid on
title('Distances Q1')
xlabel('Index nr. [-]')
ylabel('Distance [m]')

figure
set(gcf,'color','w');
plot(My_matrix(:,9),'LineWidth',2)
```

```
grid on
title('Curvatures Q1')
xlabel('Index nr. [-]')
ylabel('Curvature [-]')

figure
set(gcf,'color','w');
scatter3(Border(:,1), Border(:,2),Border(:,3),'k*','LineWidth',5)
hold on
scatter3(POV(:,1),POV(:,2),POV(:,3),'rs','LineWidth',10)
quiver3(Border9(:,1),Border9(:,2),Border9(:,3),
Border9(:,6),Border9(:,7),Border9(:,8),'r','LineWidth',3)
scatter3(min_curvature(:,1),min_curvature(:,2),min_curvature(:,3),'gd','LineWidth',20)
title('Hybrid method Q1')
xlabel('x [m]')
ylabel('y [m]')
zlabel('z [m]')

%%                    NBV - New POV calculation                    %%


n=1;

while n<=size(min_curvature,1)


    POV2 =  [min_curvature(:,1)+rho.*sin(min_curvature(:,5)).*cos(min_curvature(:,4))...
             min_curvature(:,2)+rho.*sin(min_curvature(:,5)).*sin(min_curvature(:,4))...
             min_curvature(:,3)+rho.*cos(min_curvature(:,5))];

    POVQ1 = [POV2,90-rad2deg(min_curvature(:,4)*-1),90-rad2deg(min_curvature(:,5))];

    result2 = pdist2(coor(:,:),POV2(n,:));
    flt2 = result2<= rho+sense ; %% 0.6 m working distance + 0.3 m diepte
    coor3 = coor(flt2,:);

    coorQ1 = [coorQ1;coor3];

    hold on
    scatter3(coor(:,1),coor(:,2),coor(:,3),'y.')
    scatter3(POV(:,1),POV(:,2),POV(:,3),'rs','filled')
    scatter3(POV2(:,1),POV2(:,2),POV2(:,3),'bd','LineWidth',20)
    scatter3(coor2(:,1),coor2(:,2),coor2(:,3))
    scatter3(coor3(:,1),coor3(:,2),coor3(:,3))

    n=n+1;

end

POVd = [POV,0,0];
ALL_POV = [POVd;POVQ1];

ALL_coor = [coor2;coorQ1];
ALL_coor = unique(ALL_coor, 'rows', 'stable');

figure
```

```
set(gcf,'color','w');
scatter3(coor(:,1),coor(:,2),coor(:,3),'r.')
hold on
scatter3(ALL_coor(:,1),ALL_coor(:,2),ALL_coor(:,3),'g.')
scatter3(ALL_POV(:,1),ALL_POV(:,2),ALL_POV(:,3),'bd','LineWidth',20)

figure
set(gcf,'color','w');
scatter3(ALL_coor(:,1),ALL_coor(:,2),ALL_coor(:,3))


%%                          Final results                          %%

close all               %% Comment this for opening figures at every step!


ALL_coor = [ALL_coor;AAA2];
ALL_coor = unique(ALL_coor, 'rows', 'stable');

ALL_POV_F = [NBV_Round1;ALL_POV_Fold;ALL_POV_F] ;
ALL_POV_F = unique(ALL_POV_F, 'rows', 'stable');


NBV = ALL_POV_F;
writematrix(NBV); % This step is needed for dodging a Matlab issue
% (not clearing duplicate values)
NBV = load('NBV.txt');
NBV = unique(NBV, 'rows', 'stable')      % Duplicate values are now cleared


NBV_FINAL = size(NBV,1)

figure
set(gcf,'color','w');
scatter3(NBV(:,1),NBV(:,2),NBV(:,3),'bd','LineWidth',10)

hold on
scatter3(coor(:,1),coor(:,2),coor(:,3),'r.')
scatter3(ALL_coor(:,1),ALL_coor(:,2),ALL_coor (:,3),'g.')
title(['Total NBV = ',num2str(NBV_FINAL),' || Coverage = ',num2str(coverage),'% '])

xlabel('x [m]')
ylabel('y [m]')
zlabel('z [m]')
view([0 0])

toc
```

B
Appendix B

This appendix contains the Delaunay triangulation code from reference [38] and is the work of M. Awrangjeb. He used this code in his own work in [40]. It is slightly adjusted for this thesis.

```
function [bids, E, Ne] = find_delaunay_boundary03_fig1(X,Fd)
%inputs:
%X = n by 3 array (Easting, Northing, Height) or (X,Y,Z)
%Fd = dmax (max point to point distance
%outputs
%bids: IDs to X that presents the sequence of points in the extracted
%boundary
%Please refer the paper: M. Awrangjeb, "Using point cloud data to identify,
%trace, and regularize the outlines of buildings" International Journal
% of Remote Sensing, Volume 37, Issue 3, February 2016, pages 551-579
%Open access at: http://www.tandfonline.com/doi/pdf/10.1080/01431161.2015.1131868

aThresh = 22.5; %standard 45 degree
dFd = 2*Fd;
msd = dFd*dFd;
msd1 = Fd*Fd;
TRI = delaunay(X(:,1),X(:,2));
TRI = sort(TRI,2);
%draw
figure;
triplot(TRI,X(:,1),X(:,2),'-c'); hold on;
xlim([-30,-9.5]); ylim([-32,+32]);%for A-shape


%for X
mnX = min(X(:,1));
mxX = max(X(:,1));
mnY = min(X(:,2));
mxY = max(X(:,2));
bb = 0.5; xlim([mnX-bb,mxX+bb]); ylim([mnY-bb,mxY+bb]);%for noisy shape


nP = size(X,1);
E = zeros(nP,nP) == 1;
Ne = zeros(nP,nP);%number of times an edge is shared by triangles (max 2, min 0)
Xr = [];%removed edges
for i = 1:size(TRI,1)
    T = TRI(i,:);
    Ne(T(1,1),T(1,2)) = Ne(T(1,1),T(1,2)) + 1;
    Ne(T(1,2),T(1,3)) = Ne(T(1,2),T(1,3)) + 1;
    Ne(T(1,1),T(1,3)) = Ne(T(1,1),T(1,3)) + 1;

    E(T(1,1),T(1,2)) = 1;
    E(T(1,2),T(1,1)) = 1;
    E(T(1,2),T(1,3)) = 1;
    E(T(1,3),T(1,2)) = 1;
    E(T(1,1),T(1,3)) = 1;
    E(T(1,3),T(1,1)) = 1;
end

%show boundary edges
```

```
for i = 1:nP
    for j = 1:nP
        if Ne(i,j) == 1
          plot([X(i,1) X(j,1)], [X(i,2) X(j,2)],'-m'); hold on;
        end
    end
end
plot(X(:,1), X(:,2), '.k', 'markersize', 10); hold on;
tv = [294; 295; 326; 327];
Q = [];
for i = 1:nP
    for j = 1:nP
        if (i < j && E(i,j) == 1 && Ne(i,j) == 1)% && ~V(i,j))
        % check only the upper triangle; E(i,j) = 1 indicates this is an outside edge
                sd = (X(i,1)-X(j,1))*(X(i,1)-X(j,1)) + (X(i,2)-X(j,2))*(X(i,2)-X(j,2));
            if sd > msd
                out1 = [i j]
                if i == 2986 || j == 2986
                    here = 1;
                end
                %remove this
                plot([X(i,1) X(j,1)], [X(i,2) X(j,2)],'-r'); hold on;
                Xr = [Xr; [X(i,1:2), X(j,1:2)]];
                E(i,j) = 0;
                E(j,i) = 0;
                Ne(i,j) = 0;

                if sum(i == tv) == 1 || sum(j == tv) == 1
                    here = 1;
                end
                %find other vertex k of triangle ijk and add edge ik and jk
                %as suspects
                k = find(E(i,:) & E(j,:) == 1);
                if size(k,2)>1
                    here = 1;
                    P1 = X(i,:);
                    P2 = X(j,:);
                    if (P2(1,1) - P1(1,1)) == 0
                        P1(1,1) = P1(1,1)+0.001;
                        X(i,1) = P1(1,1);
                    end
                    m = (P2(1,2) - P1(1,2))/(P2(1,1) - P1(1,1));
                    c = P1(1,2) - m*P1(1,1);
                    Pks = X(k',:);
                    num = sqrt(1 + m*m);
                    den = m*Pks(:,1) - Pks(:,2) + c;
                    dks = abs(den/num);
                    [mn id] = min(dks);
                    k = k(1,id);
                end
                if size(k,2) == 1%found
                    %for edge (i,k)
                    if i < k
                        Ne(i,k) = Ne(i,k)-1;
                        Q = [Q; [i k]];
                    else
```

```
                    Ne(k,i) = Ne(k,i)-1;
                    Q = [Q; [k i]];
                end
                %for edge (j,k)
                if j < k
                    Ne(j,k) = Ne(j,k)-1;
                    Q = [Q; [j k]];
                else
                    Ne(k,j) = Ne(k,j)-1;
                    Q = [Q; [k j]];
                end
            else%not found, error! check if happen
            end
        else

        end
    end
end

while size(Q,1)>0
    i = Q(1,1); j = Q(1,2);
    if E(i,j) == 1
        %plot([X(i,1) X(j,1)], [X(i,2) X(j,2)],'-y'); hold on;
        sd = (X(i,1)-X(j,1))*(X(i,1)-X(j,1)) + (X(i,2)-X(j,2))*(X(i,2)-X(j,2));
        if sd > msd
            out2 = [i j]
            if i == 2986 || j == 2986
                here = 1;
            end
            %remove this
            plot([X(i,1) X(j,1)], [X(i,2) X(j,2)],'-r'); hold on;
            Xr = [Xr; [X(i,1:2), X(j,1:2)]];
            E(i,j) = 0;
            E(j,i) = 0;
            Ne(i,j) = 0;
            %find other vertex k of triangle ijk and add edge ik and jk
            %as suspects
            if sum(i == tv) == 1 || sum(j == tv) == 1
                here = 1;
            end
            k = find(E(i,:) & E(j,:) == 1);
            if size(k,2)>1
                here = 1;
                P1 = X(i,:);
                P2 = X(j,:);
                if (P2(1,1) - P1(1,1)) == 0
                    P1(1,1) = P1(1,1)+0.001;
                    X(i,1) = P1(1,1);
                end
                m = (P2(1,2) - P1(1,2))/(P2(1,1) - P1(1,1));
                c = P1(1,2) - m*P1(1,1);
                Pks = X(k',:);
                num = sqrt(1 + m*m);
                den = m*Pks(:,1) - Pks(:,2) + c;
                dks = abs(den/num);
```

```
                [mn id] = min(dks);
                k = k(1,id);
            end
            if size(k,2) == 1%found
                %for edge (i,k)
                if i < k
                    Ne(i,k) = Ne(i,k)-1;
                    Q = [Q; [i k]];
                else
                    Ne(k,i) = Ne(k,i)-1;
                    Q = [Q; [k i]];
                end
                %for edge (j,k)
                if j < k
                    Ne(j,k) = Ne(j,k)-1;
                    Q = [Q; [j k]];
                else
                    Ne(k,j) = Ne(k,j)-1;
                    Q = [Q; [k j]];
                end
            else%not found, error! check if happen
            end
        else

        end
    end
    %else%this edge alreday marked unused in an iteration of the above code
        if size(Q,1)>1
            Q = Q(2:end,:);
        else
            Q = [];
        end
    %end
end
```

The remaining part of the code can be found at [38].

C
Appendix C

This appendix contains the normal vector and curvature calculation code from the reference [41].

```
function [normals,curvature] = findPointNormals(points,numNeighbours,
viewPoint, dirLargest)
%FINDPOINTNORMALS Estimates the normals of a sparse set of n 3d points by
% using a set of the closest neighbours to approximate a plane.
%
%   Required Inputs:
%   points- nx3 set of 3d points (x,y,z)
%
%   Optional Inputs: (will give default values on empty array [])
%   numNeighbours- number of neighbouring points to use in plane fitting
%       (default 9)
%   viewPoint- location all normals will point towards (default [0,0,0])
%   dirLargest- use only the largest component of the normal in determining
%       its direction wrt the viewPoint (generally provides a more stable
%       estimation of planes near the viewPoint, default true)
%
%   Outputs:
%   normals- nx3 set of normals (nx,ny,nz)
%   curvature- nx1 set giving the curvature
%
%   References-
%   The implementation closely follows the method given at
%   http://pointclouds.org/documentation/tutorials/normal_estimation.php
%   This code was used in generating the results for the journal paper
%   Multi-modal sensor calibration using a gradient orientation measure
%   http://www.zjtaylor.com/welcome/download_pdf?pdf=JFR2013.pdf
%
%   This code was written by Zachary Taylor
%   zacharyjeremytaylor@gmail.com
%   http://www.zjtaylor.com

%% check inputs
validateattributes(points, {'numeric'},{'ncols',3});

if(nargin < 2)
    numNeighbours = [];
end
if(isempty(numNeighbours))
    numNeighbours = 9;
else
    validateattributes(numNeighbours, {'numeric'},{'scalar','positive'});
    if(numNeighbours > 100)
        warning(['%i neighbouring points will be used in plane'...
            ' estimation, expect long run times, large ram usage and'...
            ' poor results near edges'],numNeighbours);
    end
end

if(nargin < 3)
    viewPoint = [];
end
```

```
if(isempty(viewPoint))
    viewPoint = [0,0,0];
else
    validateattributes(viewPoint, {'numeric'},{'size',[1,3]});
end

if(nargin < 4)
    dirLargest = [];
end
if(isempty(dirLargest))
    dirLargest = true;
else
    validateattributes(dirLargest, {'logical'},{'scalar'});
end

%% setup

%ensure inputs of correct type
points = double(points);
viewPoint = double(viewPoint);

%create kdtree
kdtreeobj = KDTreeSearcher(points,'distance','euclidean');

%get nearest neighbours
n = knnsearch(kdtreeobj,points,'k',(numNeighbours+1));

%remove self
n = n(:,2:end);

%find difference in position from neighbouring points
p = repmat(points(:,1:3),numNeighbours,1) - points(n(:),1:3);
p = reshape(p, size(points,1),numNeighbours,3);

%calculate values for covariance matrix
C = zeros(size(points,1),6);
C(:,1) = sum(p(:,:,1).*p(:,:,1),2);
C(:,2) = sum(p(:,:,1).*p(:,:,2),2);
C(:,3) = sum(p(:,:,1).*p(:,:,3),2);
C(:,4) = sum(p(:,:,2).*p(:,:,2),2);
C(:,5) = sum(p(:,:,2).*p(:,:,3),2);
C(:,6) = sum(p(:,:,3).*p(:,:,3),2);
C = C ./ numNeighbours;

%% normals and curvature calculation

normals = zeros(size(points));
curvature = zeros(size(points,1),1);
for i = 1:(size(points,1))

    %form covariance matrix
    Cmat = [C(i,1) C(i,2) C(i,3);...
        C(i,2) C(i,4) C(i,5);...
        C(i,3) C(i,5) C(i,6)];

    %get eigen values and vectors
```

```
    [v,d] = eig(Cmat);
    d = diag(d);
    [lambda,k] = min(d);

    %store normals
    normals(i,:) = v(:,k)';

    %store curvature
    curvature(i) = lambda / sum(d);
end

%% flipping normals

%ensure normals point towards viewPoint
points = points - repmat(viewPoint,size(points,1),1);
if(dirLargest)
    [~,idx] = max(abs(normals),[],2);
    idx = (1:size(normals,1))' + (idx-1)*size(normals,1);
    dir = normals(idx).*points(idx) > 0;
else
    dir = sum(normals.*points,2) > 0;
end

normals(dir,:) = -normals(dir,:);

end
'
```

D
Appendix D

The NBV results for the minimum distance method for the sphere are given in Table D.1.

Table D.1: *NBV results - minimum distance method - sphere*

Scan [-]	**x** [m]	**y** [m]	**z** [m]	ϕ [°]	θ [°]
1	0.7453	-1.4153	0.0000	26.0538	0
2	-0.7449	-1.4146	0.0503	-26.0543	1.8000
3	-0.7453	-1.4153	0.0000	-26.0538	0
4	0.7449	-1.4146	-0.0503	26.0543	-1.8000
5	0.1775	-1.3908	0.7708	7.3963	28.8000
6	-0.4381	-1.3055	0.8145	-19.4721	30.6000
7	-0.1775	-1.3908	-0.7708	-7.3963	-28.8000
8	0.0220	-1.3767	-0.8145	2.4448	-30.6000
9	0.8395	-1.0578	0.8573	36.4985	32.4000
10	-1.2368	-0.9331	0.3979	-51.2486	14.4000
11	-0.8395	-1.0578	-0.8573	-36.4985	-32.4000
12	0.7213	-1.0377	-0.9807	32.8030	-37.8000
13	1.3349	-0.8812	0.0000	54.8538	0
14	-0.0711	-0.8109	1.3772	-7.3657	59.4000
15	-0.1449	-0.8014	-1.3772	-9.3328	-59.4000
16	1.3343	-0.8808	-0.0503	54.8543	-1.8000
17	1.3706	-0.4656	0.6812	69.9668	25.2000
18	-1.0483	-0.5109	1.0953	-62.7175	43.2000
19	-1.3997	-0.5950	-0.4944	-64.5820	-18.0000
20	1.3513	-0.4538	-0.7264	70.4975	-27.0000
21	0.6784	-0.4502	1.3772	54.4912	59.4000
22	-1.5720	-0.2186	0.2005	-80.8861	7.2000
23	-0.9044	-0.3784	-1.2642	-65.4564	-52.2000
24	0.5670	-0.3771	-1.4477	54.3265	-64.8000
25	0.3245	-0.1285	1.5615	68.4000	77.4000
26	-0.3245	-0.1285	1.5615	-68.4000	77.4000
27	-0.3245	-0.1285	-1.5615	-68.4000	-77.4000
28	0.3245	-0.1285	-1.5615	68.4000	-77.4000

The NBV results for the maximum distance method for the sphere are given in Table D.2.

Table D.2: *NBV results - maximum distance method - sphere*

Scan [-]	**x** [m]	**y** [m]	**z** [m]	ϕ [°]	θ [°]
1	0.6186	-1.3896	0.4944	21.9942	18.0000
2	-0.7237	-1.3943	0.2998	-25.1471	10.8000
3	-0.7237	-1.3943	-0.2998	-25.1471	-10.8000
4	0.6186	-1.3896	-0.4944	21.9942	-18.0000
5	1.0929	-0.8368	0.8145	50.1648	30.6000
6	-1.1773	-0.8773	0.6354	-52.1536	23.4000
7	-1.1773	-0.8773	-0.6354	-52.1536	-23.4000
8	1.0929	-0.8368	-0.8145	50.1648	-30.6000
9	1.1953	-0.1026	1.0581	82.9110	41.4000
10	-1.3711	-0.1209	0.8145	-82.5648	30.6000
11	-1.3711	-0.1209	-0.8145	-82.5648	-30.6000
12	1.1953	-0.1026	-1.0581	82.9110	-41.4000
13	0.5377	-0.0679	1.5054	82.8000	70.2000
14	-0.7687	-0.0560	1.4021	-84.8974	61.2000
15	-0.7687	-0.0560	-1.4021	-84.8974	-61.2000
16	0.5377	-0.0679	-1.5054	82.8000	-70.2000
17	0.1468	-0.0329	1.5929	74.2860	84.6000
18	-0.1468	-0.0329	1.5929	-74.2860	84.6000
19	-0.1483	-0.0247	-1.5929	-76.7680	-84.6000
20	0.1468	-0.0329	-1.5929	74.2860	-84.6000

The NBV results for the minimum and maximum distance method for the sphere are given in Table D.3.

Table D.3: *NBV results - minimum and maximum distance method - sphere*

Scan [-]	**x** [m]	**y** [m]	**z** [m]	ϕ [°]	θ [°]
1	0.7453	-1.4153	0.0000	26.0538	0
2	0.4394	-1.3794	0.6812	17.1168	25.2000
3	-0.7449	-1.4146	0.0503	-26.0543	1.8000
4	-0.6186	-1.3896	0.4944	-21.9942	18.0000
5	-0.7453	-1.4153	0.0000	-26.0538	0
6	-0.6186	-1.3896	-0.4944	-21.9942	-18.0000
7	0.7449	-1.4146	-0.0503	26.0543	-1.8000
8	0.4394	-1.3794	-0.6812	17.1168	-25.2000
9	0.0228	-0.9799	1.2642	3.5574	52.2000
10	0.8066	-0.7927	1.1314	43.3310	45.0000
11	-0.0779	-1.0167	1.2328	-5.6781	50.4000
12	-1.0723	-0.8205	0.8573	-50.2102	32.4000
13	-0.0779	-1.0167	-1.2328	-5.6781	-50.4000
14	-1.0723	-0.8205	-0.8573	-50.2102	-32.4000
15	0.0228	-0.9799	-1.2642	3.5574	-52.2000
16	0.8066	-0.7927	-1.1314	43.3310	-45.0000
17	1.3349	-0.8812	0.0000	54.8538	0
18	0.6321	-0.0596	1.4684	81.6473	66.6000
19	-1.5198	-0.4001	0.2998	-74.6714	10.8000
20	-1.0153	-0.0903	1.2328	-82.4449	50.4000
21	-1.5198	-0.4001	-0.2998	-74.6714	-10.8000
22	-1.0153	-0.0903	-1.2328	-82.4449	-50.4000
23	1.3343	-0.8808	-0.0503	54.8543	-1.8000
24	0.6321	-0.0596	-1.4684	81.6473	-66.6000
25	1.4372	-0.3838	0.5890	74.1300	21.6000
26	0.0502	-0.0020	1.5992	90.0000	88.2000
27	-0.3245	-0.1285	1.5615	-68.4000	77.4000
28	-0.0125	-0.0020	1.5992	90.0000	88.2000
29	-0.3245	-0.1285	-1.5615	-68.4000	-77.4000
30	-0.0125	-0.0020	-1.5992	90.0000	-88.2000
31	1.4171	-0.3838	-0.6354	73.5877	-23.4000
32	0.0502	-0.0020	-1.5992	90.0000	-88.2000
33	0.3245	-0.1285	1.5615	68.4000	77.4000
34	0.3245	-0.1285	-1.5615	68.4000	-77.4000

The NBV results for the minimum curvature method for the sphere are given in Table D.4.

Table D.4: *NBV results - minimum curvature method - sphere*

Scan [-]	**x** [m]	**y** [m]	**z** [m]	ϕ [°]	θ [°]
1	0.2605	-1.3777	0.7708	10.5564	28.8000
2	-0.2605	-1.3777	0.7708	-10.5564	28.8000
3	-0.2605	-1.3777	-0.7708	-10.5564	-28.8000
4	0.2605	-1.3777	-0.7708	10.5564	-28.8000
5	0.3551	-0.7802	1.3509	23.2660	57.6000
6	-0.1137	-0.8063	1.3772	-9.4033	59.4000
7	-0.1137	-0.8063	-1.3772	-9.4033	-59.4000
8	0.3551	-0.7802	-1.3509	23.2660	-57.6000
9	0.0497	-0.0059	1.5992	90.0000	88.2000
10	-0.0123	-0.0039	1.5992	90.0000	88.2000
11	-0.0123	-0.0039	-1.5992	90.0000	-88.2000
12	0.0497	-0.0059	-1.5992	90.0000	-88.2000
13	0.0500	-0.0039	1.5992	90.0000	88.2000
14	0.0500	-0.0039	-1.5992	90.0000	-88.2000

The NBV results for the multiple curvatures method for the sphere are are given in Table D.5.

Table D.5: *NBV results - multiple curvatures method - sphere*

Scan [-]	**x** [m]	**y** [m]	**z** [m]	ϕ [°]	θ [°]
1	0.1775	-1.3908	0.7708	7.3963	28.8000
2	0.6186	-1.3896	0.4944	21.9942	18.0000
3	-0.4394	-1.3794	0.6812	-17.1168	25.2000
4	-0.4394	-1.3794	-0.6812	-17.1168	-25.2000
5	0.1775	-1.3908	-0.7708	7.3963	-28.8000
6	0.6186	-1.3896	-0.4944	21.9942	-18.0000
7	0.7870	-0.8606	1.0953	41.1802	43.2000
8	-0.3262	-0.8381	1.3233	-20.7145	55.8000
9	-0.3262	-0.8381	-1.3233	-20.7145	-55.8000
10	0.7870	-0.8606	-1.0953	41.1802	-43.2000
11	0.2262	-0.2650	1.5615	35.9569	77.4000
12	0.2262	-0.2650	-1.5615	35.9569	-77.4000

The NBV results for the minimum distance method for the cone are given in Table D.6.

Table D.6: *NBV results - minimum distance method - cone*

Scan [-]	**x** [m]	**y** [m]	**z** [m]	ϕ [°]	θ [°]
1	0.4939	-1.2333	0.3266	33.5859	13.5986
2	-0.4795	-1.2376	0.3360	-32.6558	13.9981
3	-0.4795	-1.2376	-0.3360	-32.6558	-13.9981
4	0.4645	-1.2421	-0.3450	31.6792	-14.3839
5	0.1074	-0.8771	0.9644	6.9689	47.5021
6	-0.0784	-0.8769	0.9673	-5.2240	47.6923
7	-0.0784	-0.8769	-0.9673	-5.2240	-47.6923
8	0.1074	-0.8582	-0.9824	6.9647	-48.6394
9	1.0202	-0.8283	0.0000	45.0000	0
10	-1.0195	-0.8285	0.0347	-44.9567	1.5048
11	-1.0195	-0.8285	-0.0347	-44.9567	-1.5048
12	1.0069	-0.8320	-0.1383	44.1683	-6.0146
13	0.6276	-0.3665	1.2889	26.5766	57.8288
14	-0.6501	-0.4101	1.2357	-27.8112	55.7043
15	-0.6501	-0.4101	-1.2357	-27.8112	-55.7043
16	0.6922	-0.4091	-1.2180	28.9952	-54.0454
17	1.2912	-0.4260	0.5871	43.2336	21.8133
18	-1.2717	-0.4230	0.6333	-42.8870	23.6310
19	-1.2717	-0.4230	-0.6333	-42.8870	-23.6310
20	0.6712	-0.2817	-1.3596	27.0641	-58.1818
21	1.3362	-0.4194	0.4993	42.6362	18.1818
22	-1.3515	-0.4216	0.4508	-42.9181	16.3636
23	-0.4676	-0.2171	-1.4758	-20.5263	-67.2727

The NBV results for the maximum distance method for the cone are given in Table D.7.

Table D.7: *NBV results - maximum distance method - cone*

Scan [-]	**x** [m]	**y** [m]	**z** [m]	ϕ [°]	θ [°]
1	0.5210	-1.2250	0.3068	35.3101	12.7602
2	-0.5078	-1.2291	0.3168	-34.4704	13.1858
3	-0.5078	-1.2291	-0.3168	-34.4704	-13.1858
4	0.4939	-1.2333	-0.3266	33.5859	-13.5986
5	0.7497	-0.7308	0.8200	35.3202	37.6964
6	-0.8218	-0.7416	0.7345	-37.5232	33.2128
7	-0.7749	-0.7343	-0.7923	-36.1487	-36.2154
8	0.6954	-0.7337	-0.8616	34.0389	-40.1289
9	0.9022	-0.3402	1.1753	33.3267	47.2727
10	-1.0071	-0.3625	1.0668	-35.8382	41.8182
11	-0.9381	-0.3481	-1.1403	-34.2079	-45.4545
12	0.8278	-0.3228	-1.2418	31.4278	-50.9091
13	0.8278	-0.3228	1.2418	31.4278	50.9091
14	-0.9381	-0.3481	1.1403	-34.2079	45.4545
15	-0.8654	-0.3317	-1.2092	-32.4003	-49.0909
16	0.7506	-0.3034	-1.3033	29.3418	-54.5455
17	0.7506	-0.3034	1.3033	29.3418	54.5455
18	-0.7895	-0.3134	-1.2732	-30.4085	-52.7273
19	-0.8654	-0.3317	1.2092	-32.4003	49.0909

The NBV results for the minimum and maximum distance method for the cone are given in Table D.8.

Table D.8: *NBV results - minimum and maximum distance method - cone*

Scan [-]	**x** [m]	**y** [m]	**z** [m]	ϕ [°]	θ [°]
1	0.4939	-1.2333	0.3266	33.5859	13.5986
2	0.2789	-1.2880	0.4153	19.2532	17.3969
3	-0.4795	-1.2376	0.3360	-32.6558	13.9981
4	-0.5972	-1.2007	0.2284	-40.0199	9.4630
5	-0.4795	-1.2376	-0.3360	-32.6558	-13.9981
6	-0.6773	-1.1710	-0.0141	-45.0911	-0.5819
7	0.4645	-1.2421	-0.3450	31.6792	-14.3839
8	0.5881	-1.2037	-0.2403	39.4653	-9.9647
9	0.0561	-0.8187	1.0229	4.5335	51.2591
10	0.7991	-0.7378	0.7638	36.9023	34.7205
11	-0.0343	-0.8379	1.0063	-3.8348	50.1858
12	-1.1309	-0.7110	0.1212	-44.2408	5.0732
13	-0.0784	-0.8769	-0.9673	-5.2240	-47.6923
14	-1.1507	-0.6971	-0.0409	-44.9567	-1.7024
15	0.1074	-0.8582	-0.9824	6.9647	-48.6394
16	1.0175	-0.7309	-0.4582	42.2739	-19.7659
17	1.3506	-0.4392	0.3967	43.8302	14.5423
18	0.5902	-0.2577	1.4103	24.5925	61.8182
19	-0.5245	-0.2979	1.3876	-22.5690	63.4736
20	-0.3845	-0.1877	1.5120	-17.0113	70.9091
21	-0.5892	-0.3552	-1.3137	-25.2912	-59.6209
22	-1.3515	-0.4216	-0.4508	-42.9181	-16.3636
23	0.5686	-0.3002	-1.3742	23.9551	-61.7237
24	1.3881	-0.4268	-0.3028	43.5790	-10.9091
25	1.3362	-0.4194	0.4993	42.6362	18.1818
26	-1.3515	-0.4216	0.4508	-42.9181	16.3636
27	-0.7895	-0.3134	1.2732	-30.4085	52.7273
28	-0.4676	-0.2171	-1.4758	-20.5263	-67.2727
29	-1.2369	-0.4041	-0.7332	-40.7407	-27.2727
30	0.6712	-0.2817	-1.3596	27.0641	-58.1818

The NBV results for the minimum curvature method for the cone are given in Table D.9.

Table D.9: *NBV results - minimum curvature method - cone*

Scan [-]	**x** [m]	**y** [m]	**z** [m]	ϕ [°]	θ [°]
1	0.0723	-1.3128	0.4441	6.0828	18.6450
2	-0.0464	-1.3143	0.4445	-4.2739	18.6645
3	-0.0464	-1.3143	-0.4445	-4.2739	-18.6645
4	0.0723	-1.3128	-0.4441	6.0828	-18.6450
5	0.0556	-0.7798	1.0576	4.5335	53.5277
6	-0.8212	-1.0243	0.0704	-44.3664	3.0577
7	-0.8212	-1.0243	-0.0704	-44.3664	-3.0577
8	0.0556	-0.7798	-1.0576	4.5335	-53.5277
9	0.7783	-0.5429	1.0224	33.2324	45.6195
10	-0.5846	-0.3662	1.3043	-25.2912	59.5111
11	-0.0163	-0.0095	-1.5998	-2.7256	-89.0909
12	0.6478	-0.3240	-1.3248	26.5766	-58.1002
13	1.3118	-0.4456	0.4897	43.2336	18.1726
14	-0.0163	-0.0095	1.5998	-2.7256	89.0909
15	0.0498	-0.0285	-1.5982	4.5335	-87.2727

The NBV results for the multiple curvatures method for the cone are given in Table D.10.

Table D.10: *NBV results - multiple curvatures method - cone*

Scan [-]	**x** [m]	**y** [m]	**z** [m]	ϕ [°]	θ [°]
1	0.1079	-1.3107	0.4418	8.0737	18.5477
2	-0.0464	-1.3143	0.4445	-4.2739	18.6645
3	-0.0464	-1.3143	-0.4445	-4.2739	-18.6645
4	0.1079	-1.3107	-0.4418	8.0737	-18.5477
5	0.3904	-0.8038	0.9631	21.7174	47.0799
6	0.4987	-0.8297	0.8850	26.8948	42.3734
7	0.5983	-0.8522	0.7948	31.3034	37.3058
8	0.8476	-0.9948	0.1012	44.1595	4.4218
9	-0.6093	-0.8948	0.7334	-32.2866	34.1022
10	-0.5651	-0.8793	0.7866	-30.1480	36.9397
11	-0.5193	-0.8614	0.8379	-28.0926	39.7618
12	-0.4678	-0.8578	0.8716	-25.8182	41.7053
13	-0.4135	-0.8394	0.9176	-22.8937	44.3992
14	-0.3606	-0.8180	0.9607	-20.3186	47.0178
15	-0.3050	-0.8130	0.9843	-17.5825	48.5495
16	-0.2474	-0.8089	1.0039	-14.4698	49.8600
17	-0.1880	-0.8056	1.0195	-11.0036	50.9264
18	-0.1298	-0.7836	1.0480	-7.5919	52.8314
19	-0.0747	-0.7810	1.0555	-4.6565	53.3718
20	-0.6093	-0.8948	-0.7334	-32.2866	-34.1022
21	-0.5657	-0.8790	-0.7866	-30.2392	-36.9397
22	-0.5193	-0.8614	-0.8379	-28.0926	-39.7618
23	-0.4678	-0.8578	-0.8716	-25.8182	-41.7053
24	-0.4138	-0.8392	-0.9176	-22.9472	-44.3992
25	-0.3606	-0.8180	-0.9607	-20.3186	-47.0178
26	-0.3050	-0.8130	-0.9843	-17.5825	-48.5495
27	-0.2478	-0.8088	-1.0039	-14.5338	-49.8600
28	-0.1880	-0.8056	-1.0195	-11.0036	-50.9264
29	-0.1298	-0.7836	-1.0480	-7.5919	-52.8314
30	-0.0747	-0.7810	-1.0555	-4.6565	-53.3718
31	0.8476	-0.9948	-0.1012	44.1595	-4.4218