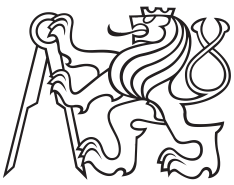
Master Thesis



Czech Technical University in Prague

F3

Faculty of Electrical Engineering
Department of Cybernetics

Automating 3D Scanning of Factory Hall by a Mobile Robot

Jakub Rozlivek

Supervisor: prof. Ing. Tomáš Svoboda, Ph.D.
Field of study: Cybernetics and Robotics
January 2022

Acknowledgements

First of all, I wish to thank Tomáš Svoboda for patiently supervising this thesis and guiding me through the process. Moreover, I would like to Tomáš Petříček for helping me with the problematics and answering all my questions.

I also wish to thank Jan Bayer for providing his robot autonomous control programs and helping with their integration. I wish to thank Tomáš Rouček for introducing me the Husky robot and the Leica scanner. More of my thanks go to Bedřich Himmel who helped me with the real-world experiments and all the hardware setup.

Especially my gratitudes go to Jana for her patience, willingness, care and help not only with this thesis. Last but not least I would like to thank my family for their continuous support.

Declaration

I declare that the presented work was developed independently and that I have listed all sources of information used within it in accordance with the methodical instructions for observing the ethical principles in the preparation of university theses.

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

V Praze, 4. ledna 2022

Abstract

These days, factories are often planned in simulation environments – digital twins first. The advantage of these digital twins is that the preparation and testing the entire robotized production line can be performed before the actual factory hall is built. The process assumes the equipment locations in the factory correspond exactly the digital twin ones. Therefore, it is necessary to check the actual locations and fix inaccuracies either in the factory or in its simulation. This work proposes a method of autonomous 3D scanning in factory halls with a mobile robotic platform. As a main part of the method, we designed a new exploration algorithm, called Exploration by Static Scans (ESS), that optimizes the completeness of 3D data needed for the installation check in combination with minimizing the time needed for the process. Due to technical limitations, the high-resolution point clouds are not available in real-time during the factory mapping process. The algorithm estimates several scanning positions based on the digital twin model. The following scanning positions are determined real-time from low-quality LiDAR data. At first, we verified the method in a model of the factory hall in a robotic simulator. Then we assembled a hardware prototype of a scanning mobile robotic platform and verified its functionality in a real factory hall environment. Finally, we summarized the obtained knowledge of using an autonomous robot to automate 3D scanning in factory halls in a study of technical feasibility.

Keywords: mobile robots, autonomous robots, 3D scanning, exploration, mapping

Supervisor: prof. Ing. Tomáš Svoboda, Ph.D.

Abstrakt

V dnešní době jsou továrny nejprve připraveny v simulovaci jako tzv. digitální dvojčata. Výhodou těchto digitálních dvojčat je, že přípravu a testování celé robotizované výrobní linky lze provést ještě před samotnou výstavbou tovární haly. Tento proces předpokládá, že umístění zařízení v továrně přesně odpovídá umístění digitálního dvojčete. Proto je nutné zkontrolovat skutečná umístění a opravit nepřesnosti buď v továrně, nebo v její simulaci. Tato práce navrhuje metodu autonomního 3D skenování v továrních halách s mobilní robotickou platformou. Jako hlavní část práce jsme navrhli nový explorační algoritmus nazvaný Exploration by Static Scans (ESS), který optimalizuje úplnost 3D dat potřebných pro kontrolu instalace v kombinaci s minimalizací času potřebného pro celý skenovací proces. Kvůli technickým omezením nejsou naskenovaná data během skenovacího procesu k dispozici. Algoritmus odhaduje několik skenovacích pozic na základě modelu digitálního dvojčete. Následující skenovací pozice jsou určeny v reálném čase z nekvalitních dat LiDAR. Metodu jsme nejdříve ověřili na modelu tovární haly v robotickém simulátoru. Poté jsme sestavili prototyp skenovací mobilní robotické platformy a ověřili její funkčnost v reálném prostředí tovární haly. Nakonec jsme získané poznatky o využití autonomního robota pro automatizaci 3D skenování v továrních halách shrnuli do studie technické proveditelnosti.

Klíčová slova: mobilní roboty, autonomní roboty, 3D skenování, explorace, mapování

Překlad názvu: Automatizace 3D skenování tovární haly pomocí mobilního robotu

Contents

Figures

Tables

Chapter 1

Introduction

1.1 Motivation

These days, factories are often planned in simulation environments – digital twins first.The advantage of these digital twins is that the preparation and testing the entire production line can be performed before the factory is built. The process assumes the equipment locations in the factory correspond exactly the digital twin ones. Otherwise, it is necessary to program the robots when deployed to avoid possible damage caused by their collision with other objects. This would result in manufacturing downtime and financial loss. For these reasons, it is necessary to verify the equipment positions in the factory after or better during the factory construction, and fix occurred inaccuracies either in the factory or in its simulation. Nevertheless, factories are often large; thus, scanning is time-consuming. Moreover, it has to be done when the production is suspended, such as during nights, weekends, or planned downtime; hence, it has to be planned in advance. Because of these problems, it is desirable to automate the scanning process and let a robotic platform make scans whenever needed without dependence on a human operator.

This task requires robust autonomous control of the robot and effective and complete environment coverage. However, current environment exploration methods aim to effectively explore environment as fast as possible without coverage density constraints and with a precision of several centimeters. Nevertheless, this is inadequate for precise factory mapping. Therefore, we would like to adapt these methods to this specific task.

1.2 Goals

The main goal of this thesis is to propose a method of autonomous 3D scanning in factory halls with a mobile robotic platform supplemented by a recommendation of an appropriate hardware setup. An essential part of this goal is to introduce a new exploration algorithm that optimizes the completeness of 3D data needed for the installation check, i.e., checking the position and orientation of individual robot manipulators and robotic cells during the construction or reconstruction of the factory arrangement. We will determine how to profit from the gained experiences with exploration of an unknown underground terrain. Our exploration algorithm must incorporate the constraint that the scanner must stand still when scanning, which lasts several minutes. Moreover, it is necessary to approximate the scans gathered from this scanning procedure using data gathered by a laser scanner (LiDAR) since the data from the high-resolution scanner are not available during the factory mapping process. We will find a criterion function that prefers high-density coverage of the environment instead of maximizing the explored volume. Furthermore, we will determine whether and how to exploit the prior information about the environment to improve and speed up the whole

process.

We will assemble a prototype of hardware solution from available robotic and sensory equipment, and test it in a real factory hall environment. We will propose solutions to detected real-world problems. Furthermore, we have to find an optimal way of sensor data processing during the whole procedure to have as much information about the environment as possible, considering time and memory limits.

Finally, we will conduct a study of the technical feasibility of using an autonomous robot in combination with the proposed exploration algorithm to automate 3D scanning in factory halls. We will assess whether our current autonomous robot control is robust, safe, and precise enough for this task. Eventually, we have to find improvements that have to be done to ensure it, e.g., place calibration markers in the environment to improve the robot's localization.

1.3 Related work

Recently, as computing power has been improved and sensors used to obtain 3D information, such as cameras and laser scanners, have expanded significantly in range, accuracy, and affordability, the 3D modeling of environments became popular. Without a doubt, one of the industrial areas where this technology will benefit is engineering and construction, where increasing attention is paid to quality control and control processes. Since the construction sites are usually extensive, an effective planning algorithm for data acquisition in an unknown environment is essential. Klein et al. [1] proposed one of the first next-view-based algorithms for large complex environments. Adan et al. [2] provided an overview of existing autonomous scanning systems and discussed their practical applicability. They call the problem of the next best location decision as the Next-Best-Scan problem. Similar to [3], they group the solution strategies into two categories — Frontier-based and Information-based. An example of the frontier-based approach is presented in [4]. They proposed an automatic planning method for efficient and accurate 3D modeling of the environment with the help of several robots. This method replaces the robot localization based on the iterative closest points algorithm with a multi-robot localization technique called Cooperative Positioning System. The system consists of the parent robot carrying a laser sensor and several child robots (wheeled robots and quadcopters) carrying light white balls for identification.

Strategies exploiting Information Gain estimate the next best view based on how much of the unknown volume is visible from that view. Several examples of Information Gain computation are presented, in e.g., [3, 5, 6]. Kriegel et al. [6] maximized the object surface coverage instead of minimizing unseen area as it is done in [3, 5]. Bissmarck et al. [7] compared the effectiveness of several different approaches to Next Best View evaluation for both indoor and outdoor environments. They realized that the differences in accuracy and efficiency were insignificant and that the main difference was in performance time. The HRT (Hierarchical ray tracing) algorithm [8] achieved the best time results. However, this method does not use the Information Gain computation, unlike the other compared methods. Therefore, the authors proposed a new algorithm FVHRT (Frontier oriented volumetric hierarchical ray tracing), with a similar algorithm speed as HRT but with Information Gain metric.

The autonomous scanning approaches can be divided according to the way how the scans are collected—simultaneously while moving or only when the robot stands still (i.e., stop-and-go

procedure). The first variant is often preferred for aerial robots as in [9, 10]. Meng et al. [10] proposed a new two-stage method of autonomous reconstruction of the 3D environment tested on drones simultaneously scanning with LiDAR. In the first step, suitable candidates to visit are selected based on their Information Gain. In the second step, the optimal order of visiting these points is calculated to maximize coverage of the space. Subsequently, the trajectory to the first point is calculated. Then, the whole process is repeated, due to the dynamically changing map, until the space is completely covered.

On the other hand, the second variant is often tested on ground robots, e.g., in [11, 12, 13] or underwater robots [14]. Palomeras et al. [14] proposed a probabilistic method of automatic planning of individual scanning positions (next-best-view algorithm) to map and inspect complex underwater environments. Their strategy selects scanning position candidates around the current position to avoid moving back and forth.

Blaer and Allen [11] proposed a method for automatic planning of individual scanning positions to reconstruct 3D models of outdoor and indoor spaces. The procedure is divided into two phases. In the first phase, a set of initial scanning positions is determined from the prior 2D model to cover as much space as possible (also known as the art gallery problem). The ordering of these scanning positions is formulated as a traveling salesman problem. An initial 3D model is created from these scans. In the second phase, a set of scanning position candidates is generated from a border between known and unknown parts of the environment. The next scanning position is the candidate with the highest expected contribution to the 3D model. This process is repeated until the model improvement is below a threshold.

Kim et al. [12] utilize flexible 2D SLAM for robot position estimation in 3D. Their robotic platform contains two LiDARs, one for dynamic scans and one for high-resolution scans for an accurate 3D environment model. Dynamic scans are performed as the robot moves through the environment. The data from dynamic scans are used to identify obstacles and suitable locations for high-resolution static scans. The algorithm computes a fitness score for each candidate along the robot trajectory based on the occlusion.

Similar to the other NBV methods, the method proposed in [15] selects the next scanning position based on minimizing the number of occluded voxels. They introduced three new types of voxels (*window*, *door*, and *out*), aside from the usual three types of voxels (*empty*, *occupied*, *occluded*). Their method locates the doors and windows in the scanned room model and removes them from the model. Moreover, the method exploits the geometry of the room for the registration of point clouds.

1.4 Contribution

The contributions of this thesis are listed as follows:

- We designed a new exploration algorithm called Exploration by Static Scans (ESS) optimizing the completeness of 3D data needed for the installation check in combination with minimizing the time needed for the process.
- We proposed an approximation method for estimating the gathered high-resolution point clouds from terrestrial scanner which are not available real-time during the factory mapping process. The approximation exploits the information about the environment

gathered by the LiDAR.

- We assembled a hardware prototype of a scanning mobile robotic platform and verified its functionality in a real factory hall environment.
- We summarized the obtained knowledge of using an autonomous robot to automate 3D scanning in factory halls in a study of technical feasibility.

Chapter 2

Experimental platform description

This chapter presents the experimental scanning mobile robot platform prepared profiting from the experience of CTU-CRAS-NORLAB robotic team [16] in DARPA SubTerranean Challenge [17]. At first, the meta-operating system and robotic simulator are presented in Sec. 2.1 followed by the scanning platform description in Sec. 2.2 and its simulation variant in Sec. 2.3. Then the solutions for robot localization and robot navigation are described in Sections 2.4 and 2.5, respectively. Finally, the original exploration algorithm is described in Sec. 2.6.

2.1 Robot Operating System and Ignition

Robots in both simulation and real world run under the Robot Operating System (ROS). ROS [18] is an open-source, meta-operating system for robots. It contains services typical for a classical operating system like hardware abstraction, low-level device control, and message-passing between processes. ROS is a collection of tools, libraries, and conventions that enable a collaborative environment for developing complex robotics software across different robotic platforms.

The simulations were prepared in a robotic simulator called Ignition Gazebo because this simulator was used in DARPA SubTerranean Challenge. Ignition [19] is a collection of open-source software libraries and cloud services mainly aimed at robot developers, simplifying high-performance applications development. Ignition is the successor of the robotic simulator Gazebo Classic. Instead of upgrading the Gazebo Classic, a collection of loosely coupled libraries was created. The following ones were used during the simulations:

- Ignition Physics — provides an interface for physics engines designed for various applications and with a range of features.
- Ignition Rendering — provides an interface for different rendering engines. It offers a unified API for creating 3D graphics applications.
- Ignition Sensors — provides various sensor models generating realistic data from simulations. It relies on other Ignition libraries, especially those providing rendering (Ignition Rendering) and physics simulation (Ignition Physics).
- Ignition Gazebo — a robotic simulator with many features, such as high fidelity physics (Ignition Physics), rendering (Ignition Rendering), and sensor models (Ignition Sensors). It supports including physical objects as meshes in one of several 3D model file types: .dae, .obj, etc.

2.2 Real-world scanning platform prototype

The experimental scanning mobile robot platform used in simulation and real-world environments was assembled regarding the specific requirements for our task. Since very precise positioning of the factory equipment is required to prevent possible robot collisions, only terrestrial scanners can be considered for high-resolution scanning. Unfortunately, the scanning procedure with this type of scanner lasts several minutes, and the scanner must stand still during the procedure. Therefore, the robot carrying the scanner has to be able to hold a stable position for several minutes. Moreover, the data from the terrestrial scanners are not available during the factory mapping process. Hence, the robot must have other sensors to move successfully in an unknown environment, which implies that its payload capacity has to be sufficient to hold all necessary sensors. Additionally, we assume that the environment is easily accessible and only in one height level. For these reasons, we chose a four-wheeled mobile robot equipped with a high-resolution terrestrial scanner for factory mapping and other sensors (e.g., LiDAR, RGBD camera, inertial measurement unit) for robot localization and navigation.

2.2.1 Robotic platform

Four-wheeled mobile robot Husky [20] (see Fig. 2.1) was selected for the factory mapping task because its construction enables holding a stable position for several minutes, and its payload capacity is sufficient for all needed sensors. Furthermore, it can traverse the environment in a reasonable speed of at up to 1 m/s with a maximum acceleration of 3 $\mathrm{m\,s^{-2}}$. Since it is designed for challenging real-world terrain, it should easily operate in a factory environment, except for the narrow sections, where omnidirectional four-legged robots like Spot [21] would be more suitable choice. However, the stable position with attached heavy payload and the robustness are the reasons why we preferred Husky robot for our experiments. Nevertheless, the usage of Spot for this task will be considered in future. Another important aspect is the robot endurance, which was updated to several hours. The robot is equipped with a number of sensors, such as several RGBD cameras, an inertial measurement unit, and a WiFi connection.

2.2.2 Terrestrial scanner

The terrestrial scanners are suitable for our task because they offer high resolution and precision (both in the magnitude of millimeters). Specifically, we chose Leica BLK360, which offers 3 user-selectable resolution modes — 5/10/20 mm @ 10 m (corresponding to an angular resolution of 0.03°/ 0.05°/ 0.11°) in both directions. The precisions presented in the datasheet are the 3D point accuracy of 6 mm @ 10 m and the measured range accuracy of 4 mm @ 10 m. The scanner can measure objects lying at a distance of from 0.6 m up to 60 m and produces around 350 thousand points per second.

The scanner is meant to be operated by a human the whole time and therefore the scanning procedure has to be started manually. It can be controlled via a tablet/smartphone application over WiFi. Another way is to use the button on the scanner. The button behavior (what should be captured and how) has to be set in the Windows or tablet/smartphone application.

2.2.3 LiDAR sensor

As scanning with the terrestrial scanner lasts several minutes and the gathered data are unavailable during the factory mapping process, the robot localization estimation and navigation have to be secured by other sensors. Moreover, it would be useful to have a possibility to estimate the data measured by the terrestrial scanner. Therefore, we decided to use a LiDAR sensor as it scans continuously with a scanning frequency of 10 Hz regardless of whether the robot is moving or not. The resolution and precision are significantly lower than for the terrestrial scanner but sufficient for the robot navigation and localization. Specifically, the chosen LiDAR Ouster os-0 has the vertical angular resolution of 0.7° and the horizontal resolution of 0.2°, which means there are six times more rays from the terrestrial scanner than from the LiDAR in the horizontal plane. The LiDAR can produce more than $2.5 \cdot 10^6$ points per second in the measurable distance from 0.3 m up to 50 m, with measurement precision from 1.5cm to 5cm depending on the distance from the sensor.

2.2.4 Custom holder for the terrestrial scanner

We decided to mount the terrestrial scanner above the sensor rack because the sensor rack with RGBD cameras and Ouster LiDAR cannot be moved, otherwise, the calibration of the rack precise position would be needed. Since the 3D printed sensor rack has holes with atypical dimensions, a custom 3D printed adapter (see Fig. 2.2 left) had to be designed to connect the sensor rack and the aluminium profiles creating the construction. The construction has a shape of the letter T with perpendicular profiles in each vertex (see Fig. 2.2 right) and a holder for the terrestrial scanner in the intersection of two profiles (center of T). The robot with the attached construction and the terrestrial scanner can be seen in Fig. 2.1.

2.3 Scanning platform in simulation

For simulation, we updated Husky robot model prepared for DARPA SubTerranean Challenge [22]. The sensor rack defined in the model specification was adjusted to match the customizations made on the real robot. The original LiDAR sensor model was derived from Robosense RS-LiDAR-16. Therefore, it was necessary to update the parameters to model Ouster os-0 LiDAR.

Like in the scanning prototype, the Leica scanner model was placed above the LiDAR in the sensor rack, but without the aluminum construction to simplify data gathering by the LiDAR, i.e., to prevent occlusion caused by the construction.

Protocol buffers [23] used in Ignition Sensors plugins limit the size of data obtained from a measurement simulation. The limit is too low to simulate the parameters similar to the Leica ones with one instance of the scanner model. Therefore, we substituted the Leica scanner with 12 instances, each of them covering a distinguished circular sector in the horizontal plane with a central angle $\theta = 30°$. The scanner model has the same vertical field of view as Leica BLK360 scanner, and the number of channels is 3000, corresponding to the medium settings in Leica BLK360 (angular resolution of 0.05°). The horizontal angular resolution is 0.05° too, and the horizontal field of view is 30° as mentioned before. Scans are generated and processed separately.

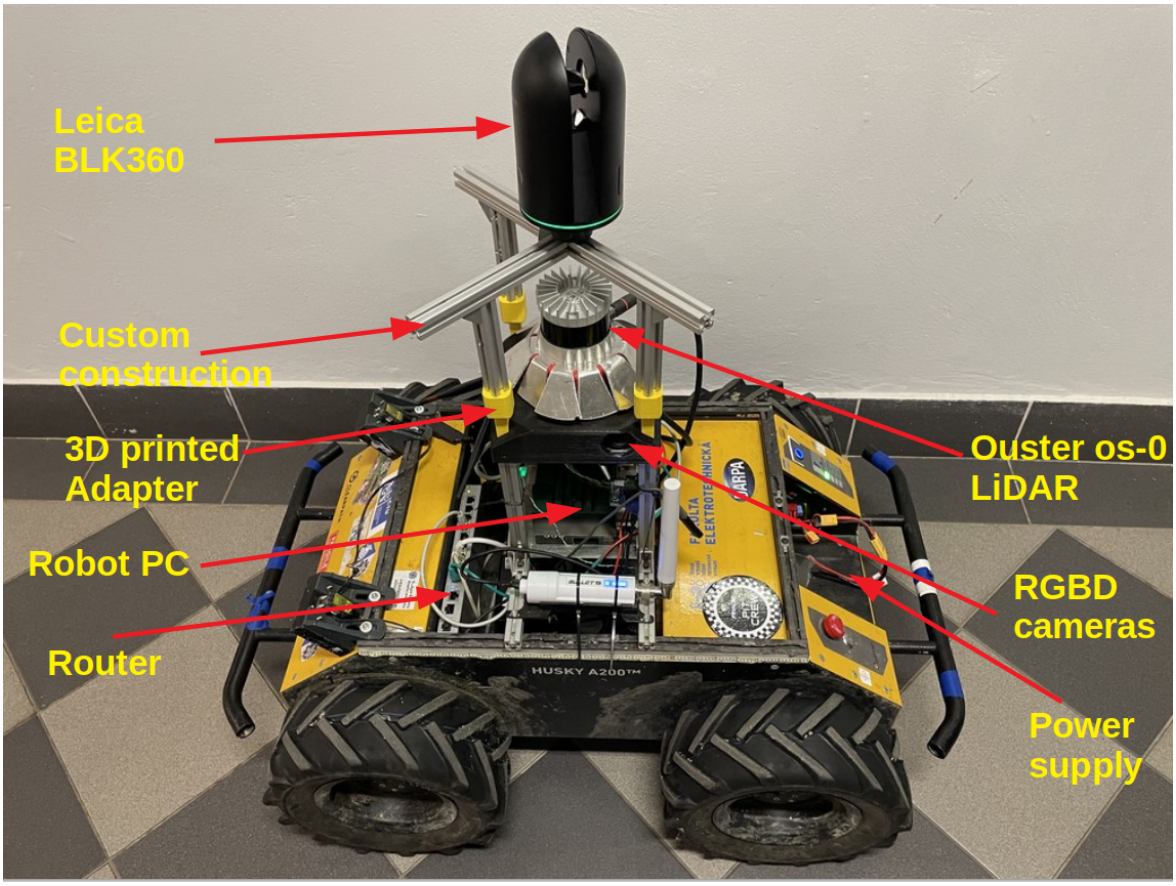


Figure 2.1: Husky robot with the attached terrestrial scanner.

2.4 Robot localization

The estimation of the precise robot position in a map, which is crucial for autonomous robot control, is ensured by a 2-D/3-D mapping library relying on the ICP algorithm called norlab-icp-mapper [24]. The principle of a basic version of the ICP algorithm is described as follows.

2.4.1 Iterative closest point algorithm

Iterative closest point algorithm by Besl and Mckay [25] is one of the most popular approaches for point cloud registration — the process of finding the spatial transformation (rotation and translation), i.e., aligning a point cloud with the other one.

The algorithm assumes two roughly aligned point clouds P and Q having an arbitrary number of 3D points. The outputs of the algorithm are rotation matrix R and translation vector t representing the transformation needed to align P with Q. The rotation matrix is initialized as identity matrix and the translation vector as zero vector.

The algorithm iteratively updates the transformation aligning point cloud P with point cloud Q to minimize the distance between the corresponding points. It can be divided into four steps:

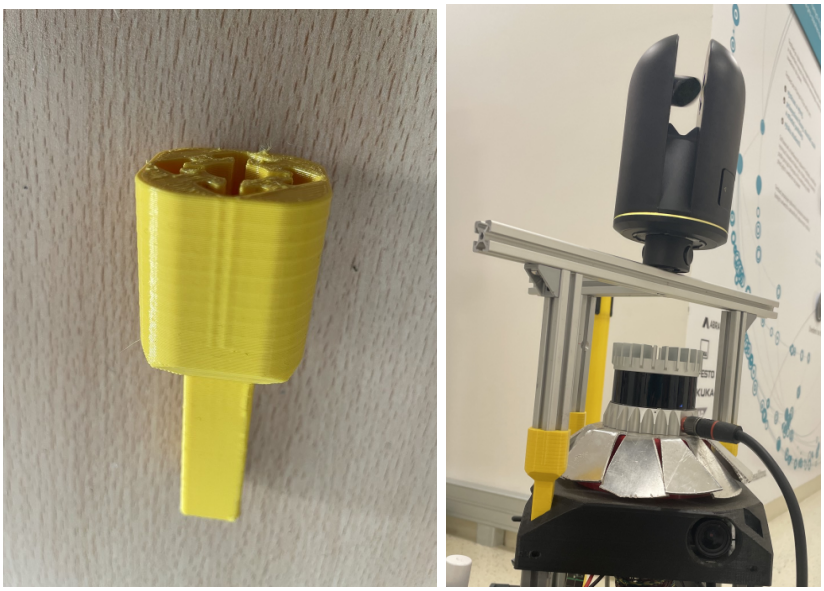


Figure 2.2: 3D printed adapter (left) and the construction attached to the sensor rack (right).

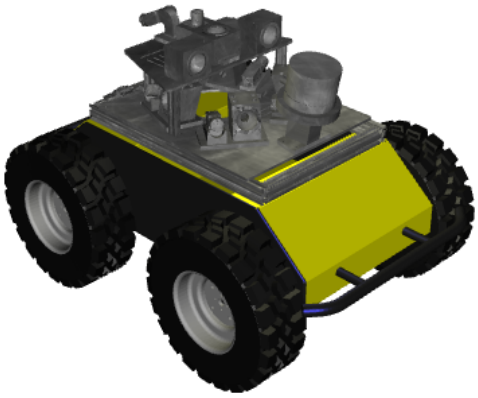


Figure 2.3: Husky robot model for Ignition Gazebo simulator.

1. For each point p_i from P, the closest point q_i in Q is found. KD-tree structure is applied to speed up the computation.

2. Correspondences between distant pairs of points are removed based on the threshold, .e.g, a median of distances.

3. Transformation minimizing distance between the corresponding points is calculated as

$$\mathbf{R}', \mathbf{t}' = \underset{\mathbf{R}' \in SO(3), \mathbf{t}' \in \mathcal{R}^3}{\operatorname{argmin}} \sum ||\mathbf{R}'\mathbf{p_i} + \mathbf{t}' - \mathbf{q_i}||_2^2. \tag{2.1}$$

This problem is solved using SVD decomposition as proposed by Arun et al. [26]:

$$\mathbf{p}'_\mathbf{i} = \mathbf{p}_\mathbf{i} - \frac{1}{N}\sum_i \mathbf{p}_\mathbf{i} = \mathbf{p}_\mathbf{i} - \tilde{\mathbf{p}}, \tag{2.2}$$

$$\mathbf{q}'_\mathbf{i} = \mathbf{q}_\mathbf{i} - \frac{1}{N}\sum_i \mathbf{q}_\mathbf{i} = \mathbf{q}_\mathbf{i} - \tilde{\mathbf{q}}, \tag{2.3}$$

$$\mathbf{H} = \sum_i \mathbf{p}'_\mathbf{i}\mathbf{q}'^T_\mathbf{i} \tag{2.4}$$

$$\mathbf{H} = \mathbf{U}\mathbf{S}\mathbf{V}^T \tag{2.5}$$

$$\mathbf{R}^* = \mathbf{V}\mathbf{U}^T \tag{2.6}$$

$$\mathbf{t}^* = \tilde{\mathbf{q}} - \mathbf{R}^*\tilde{\mathbf{p}} \tag{2.7}$$

4. Points from P are transformed, and the rotation matrix and translation vector are updated:

$$\mathbf{p}_\mathbf{i} = \mathbf{R}'\mathbf{p}_\mathbf{i} + \mathbf{t}' \tag{2.8}$$

$$\mathbf{R} = \mathbf{R}'\mathbf{R} \tag{2.9}$$

$$\mathbf{t} = \mathbf{R}'\mathbf{t} + \mathbf{t}' \tag{2.10}$$

5. Go back to Step 1 and repeat the procedure until the sum of Euclidean squared errors decreases under a specified threshold.

2.4.2 ICP-SLAM

While moving in an unknown environment, the robot has to construct a map and localize itself in the map. This problem is known as Simultaneous Localization and Mapping (SLAM). In this work, we assume an online SLAM problem, where past robot poses are not updated. The robot builds its map from the 3D LiDAR scans, aligned using ICP algorithm, therefore the method is called ICP-SLAM, and it works as follows.

Each new scan is uniformly downsampled. The remaining points are initially aligned with the robot map based on the last transformation and odometry fused from encoders in wheels and an inertial measurement unit. Then the ICP algorithm finds a more precise alignment of the point clouds to improve the localization accuracy. The aligned scan points are added to the map and the robot position is updated.

The ICP-SLAM suffers from several issues. It does not work when an incompatible combination of environment and movement appears, e.g., forward motion in a long hallway or rotation in a circular room. ICP algorithm needs a good initialization, i.e., good initial alignment estimation or small change between scanning poses, to avoid the convergence to a local minimum. Moreover, since the algorithm estimates the robot position sequentially, the position error is accumulated in time.

2.5 Robot navigation

The robot is controlled by programs from a framework Robot deployment system (further referred to as RDS) [27]. The approach [28] provides basic navigation functionality. The

framework is mainly designed for multi-robot exploration of unknown underground large-scale environments and provides several exploration strategies targeted for low bandwidth communication [29]. It can use various robot platforms, including walking robots, tracked and wheeled vehicles, and it is compatible with different driving systems; moreover, it works in various simulators.

2.5.1 Path planning

The input to the RDS framework is a sequence of scans from the LiDAR, which are used to build a local dense 3D grid map. The local grid map is used to assess terrain traversability, which is based mainly on measuring terrain roughness by the difference in the height of the neighboring cells. Besides the traversability assessment, the local map serves for robot navigation to nearby waypoints within the local map. The path from the current robot location is planned using Dijkstra's algorithm [30], which runs on a graph connecting neighboring traversable cells, and the cost of each edge is evaluated by computing distance from obstacles as shown in [31]. An example of the path planned within the local map with assessed traversability is shown in Fig. 2.4.

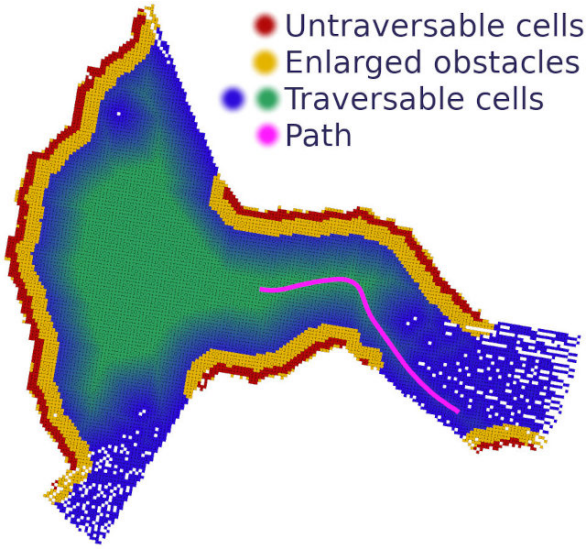


Figure 2.4: Example plan within the map with the assessed traversability. (Courtesy of [28].)

2.5.2 Navigation using the planned path

A separate module executes the path planned within the local map. The module selects the next navigational waypoint lying at least 0.5 m ahead of the robot. Then the module calculates the distance from the current robot position to the waypoint and the angle required to turn the heading of the robot to the waypoint. Finally, the forward and angular velocities are calculated based on the proportional control law.

2.6 Original frontier-based exploration

Originally, the exploration was secured by a ROS package for robot navigation and exploration called Naex [32]. The package contains two main programs — a planner that plans paths globally and a follower that follows the planned path.

The planning program either receives the next navigation goal or computes it as a part of an exploration strategy maximizing the reward/cost ratio, where frontiers are preferred. A frontier is a cell (voxel) that separates known and unknown regions. A path to the closest reachable point to the specified goal is planned if the goal is valid. If a start position is not given, the path is planned from the robot's current position. A point map is built internally from input point clouds to assess traversability and enable global path planning.

The follower program follows published paths. At each control step, the nearest point on the path is selected as a (local) navigation goal at each checkpoint. Once the path is completed or a timeout is exceeded, a new one is taken. If there is no valid path, the robot can backtrack to its previous checkpoint. The follower program checks possible collisions with objects that are detected in the input point clouds and tries to avoid them.

Chapter 3

Exploration that maximizes quality of the 3D scans

This chapter introduces the proposed exploration algorithm called Exploration by Static Scans (ESS). At first, the used data structure and mapping library are presented in Sec. 3.1. Then the algorithm is presented. It consists of two phases: Prior Knowledge (PK) and Next-Best-Scan (NBS) which are described in Sections 3.3 and 3.4, respectively, and the algorithm overview is visualized in Fig. 3.1. High-level robot control and the teleoperation are described in Sec. 3.5. Section 3.6 introduces several metrics for measured data evaluation and ESS algorithm variants comparison.

3.1 Octree data type and OctoMap library

The scanned environment is stored in **octree** [33]—a hierarchical data structure for a spatial subdivision. Model volume is recursively subdivided into eight octans and the created tree branches can be pruned at any level, if all node children have the same value and the octree values are discretized (the represented space can be either occupied or free) without any probabilistic approach.

OctoMap library [34] is an efficient probabilistic 3D occupancy grid mapping approach based on octrees, providing data structures and mapping algorithms particularly suited for robotics. OctoMap maps are stored efficiently and can model arbitrary environments without prior assumptions. It distinguishes occupied areas, free space, and unknown areas, which are encoded implicitly. The map is updated in a probabilistic fashion when new information is added and dynamically expanded when needed. Created models can be easily visualized with OctoMap visualization tool called Octovis [35].

3.2 ESS Algorithm overview

The goal of the algorithm is to find optimal Scanning positions (SPs) to maximize the coverage of the unknown environment. The current hardware platform contains two scanners, a LiDAR for robot mapping and navigation (i.e., dynamic scans) and a terrestrial scanner for the output scans (i.e., static scans). A similar approach is in [12]. As we assume that the data from the terrestrial scanner are not available during the factory mapping process, it is necessary to estimate the proportion of the environment the terrestrial scanner has already scanned. Therefore, two models of the environment were maintained. The first one consists of the dynamic scans (further referred to as dynamic model), and the second one (further referred to as static model) is estimated from approximations of static scans for which the dynamic model is used. The models are represented by an octree structure from OctoMap library. The origin of the models is in the robot starting location, and the resolution of the octree models is chosen to meet memory and time limits; usually, the voxel edge length of 0.1 m was used.

SPs are positions where the static scans are gathered. Scanning position candidates (SPCs) are sampled in the current position neighborhood (similar approach as in [14]). The algorithm may exploit the prior knowledge about the environment (if available) to speed up the exploration procedure by computing several SPs ahead (i.e., before the exploration starts) as in [11]. It is assumed that the starting position is marked to enable the precise processing of the SPs.

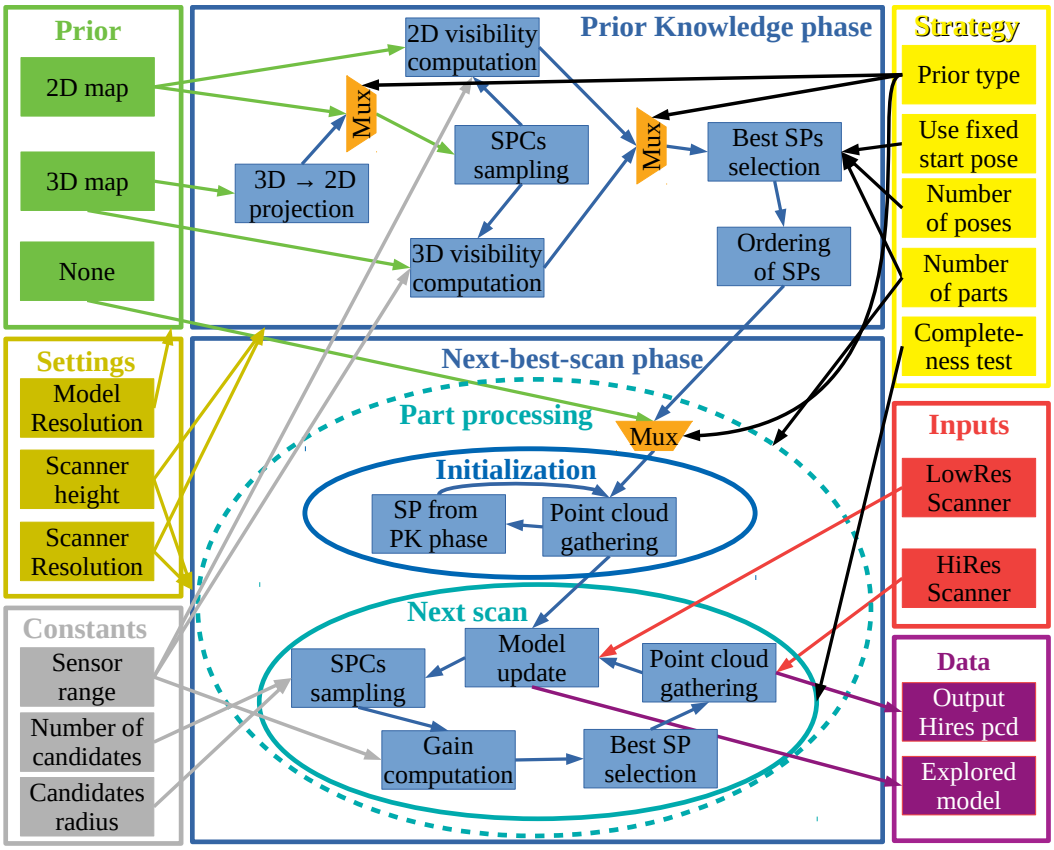


Figure 3.1: The proposed algorithm overview. The NBS phase can work with individual parts of the model separately (inside of the dashed ellipse). Prior knowledge types are marked in green. If there is no prior knowledge or the PK phase is completed, the NBS phase starts. Scanners measurements are marked in red, algorithm outputs are marked in violet, and light yellow represents variants (strategies) of the algorithm. Scanner parameters and model resolution are marked in gold, and other parameters currently set to be constant are marked in gray.

3.3 Prior Knowledge phase

The purpose of the PK phase is to exploit the prior knowledge of the environment to prepare SPs maximizing the coverage of the environment and minimizing their count so the whole procedure is faster. As the first step, the prior model is preprocessed, projected to 2D grid, in order to consider SPCs in 2D space only, and then the possible SPCs are generated (see Sec. 3.3.1). Depending on the model type, visibility for each SPC is computed either in 2D (see Sec. 3.3.2) or 3D (see Sec. 3.3.3). The best SPs are the SPCs with the highest coverage of the environment according to computed visibility (see Sec. 3.3.4). Once the best SPs are

chosen, the path between them and their order is determined (see Sec. 3.3.5). The ordered SPs and their connections are the output of the PK phase.

3.3.1 Scanning position candidates sampling

Since we assume the floor in the factory is not sloped,which means the sensor height is fixed, the SPCs are sampled in 2D space. If the prior model is in 3D, it has to be projected into 2D binary occupancy grid (cells can be either empty or occupied). The 3D model is projected from a given range of z coordinate values. If any voxel in the given range is occupied, its corresponding cell in the 2D grid is also occupied. Otherwise, it is set as empty. In our experiments, the range minimum and maximum values are 0.1 m and 0.9 m — the lower limit is higher than zero because the robot can run over small obstacles, and the upper limit is associated with the height of the robot. The scanner can be placed only in cells marked as free, which represent a safe space for the robotic platform with attached scanner(s).

Since the robot occupies more than one cell, the SPs close to the obstacles are infeasible due to possible collision. Therefore, the obstacles are inflated to add a safety margin between the robot and the obstacle. The inflation is performed by applying a morphological dilation to the grid.

Once the occupancy grid is prepared, it is subsampled uniformly with a given step between cells. Then each of these new samples is disturbed by adding a random offset to both coordinates. If the new position is empty, it is checked whether it is reachable from the robot starting position (i.e., a path between the positions exists), then it is added to the list of SPCs; otherwise, it is disturbed and checked again (five times at maximum). Example of sampled SPCs and the difference between original and inflated grid can be seen in Fig. 3.2.

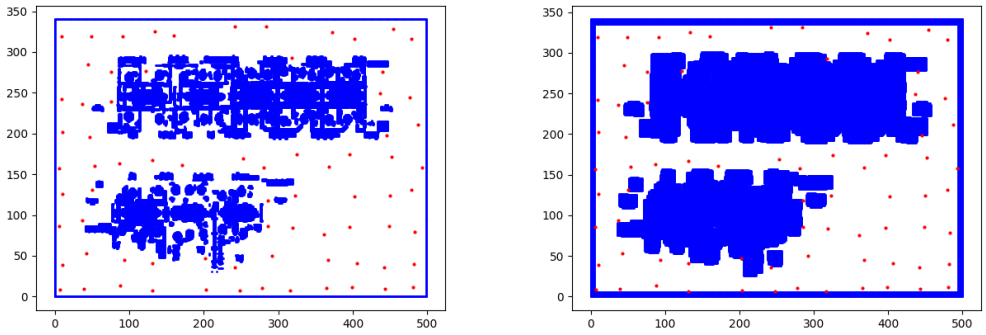


Figure 3.2: 2D grid projection of the environment with SPCs (in red) and obstacles (in blue) before (left) and after inflation (right).

3.3.2 2D visibility computation

The 2D visibility for a SPC is determined by casting rays in the 2D prior model represented by a 2D grid (the version without inflated obstacles). To simulate the scanner with a horizontal field of view of 360° in 2D, we sample points on the circle circumference as endpoints of the rays. The circle has a radius of the scanner maximum range. The points on a circle circumference can be generated in two different ways.

1) Bresenham's circle algorithm [36] — An effective form of a midpoint circle algorithm [37] calculating only one-eighth of the circle circumference. The rest of the circle is obtained by mirroring and flipping of the generated points.

2) Angle increment — This approach takes the radius R and angular resolution as input and uses the resolution to sample the values of the angle $\theta \in [-180°,\ 180°)$. Then the points on the circle can be computed as

$$x = \lfloor R \cos\theta \rfloor , \tag{3.1}$$
$$y = \lfloor R \sin\theta \rfloor , \tag{3.2}$$

where $\lfloor x \rfloor$ symbolize the floor function. The first approach covers the circle circumference more densely; however, it is not suitable for higher radius values since it consumes too much time. Therefore, the second option is preferred. Once the points on the circle are generated, the rays from the SP to the points of circumference are computed using Bresenham's line algorithm [38].

We precomputed the ray paths beforehand to accelarate the procedure. We exploited the "linearity" and symmetry and computed the rays for a circle with a center position (0,0). During the visibility computation, the points are obtained by adding the computed points to the SP coordinates.

The actual visibility is computed by iterating over the rays. For each ray, its points are traversed until the first occupied point in 2D grid is reached. All traversed points are stored in the set of visited cells. However, if the occupied cell is closer to the SP than a given scanner minimum range so the scanner could not see it, the points are not added to the set.

The output of the 2D visibility computation is a set of visited cells for the specific SPC. In this case, not only visible obstacles but all the visited cells are important to compare the SPCs except for the close neighborhood of the robot due to occlusion caused by the robot body. Example of the visited 2D points is in Fig. 3.3.

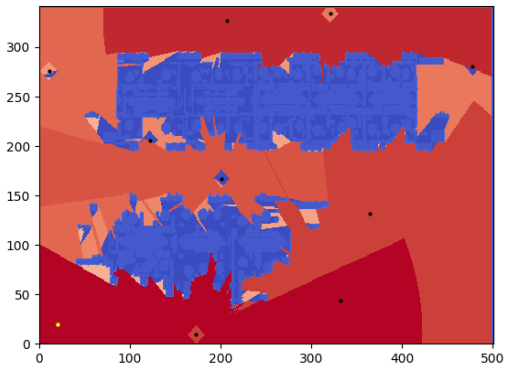


Figure 3.3: 2D visibility estimation from several SPs. Color represents individual SPs, i.e., points with the same color are seen from the same SP and blue color represents unvisited (unseen) space.

3.3.3 3D visibility computation

The 3D visibility for a SPC is computed by casting rays in the 3D prior model. The rays were cast in a space represented by an octree using the OctoMap library.

As opposed to the 2D case, it is necessary to simulate both fields of view — horizontal and vertical. Therefore, we sample points on a sphere. The points are generated by the Cartesian product of the sampled θ (horizontal angle) and ϕ (vertical angle) values. $\theta \in [-180°,\ 180°)$, and ϕ is set based on the vertical field of view. Since the ray-casting function in OctoMap takes a direction vector instead of the ray endpoint, the direction vectors are sampled on a unit sphere as

$$x = \sin\phi \cdot \cos\theta \tag{3.3}$$
$$y = \sin\phi \cdot \sin\theta \tag{3.4}$$
$$z = \cos\phi \tag{3.5}$$

and they can be precomputed ahead.

For each direction, the ray is cast with the OctoMap function castRay, which uses algorithm proposed by Amanatides and Woo [39] to traverse the voxels. If the ray reaches an obstacle, it returns coordinates of the intersection, which is added to the set of visible occupied 3D points.

A special case occurs when we want to simulate the high-resolution terrestrial scanner. The rays are cast from all model voxels in the neighborhood limited by a scanner maximum range. If the ray intersects the SP, the corresponding model voxel is stored in the set of visible occupied 3D points. This method guarantees that all the model voxels visible from the SP will be stored in the set.

The output of the 3D visibility computation is the set of visible 3D points, i.e., a gathered point cloud, for the specific SPC. Example of the visible 3D points is in Fig. 3.4.

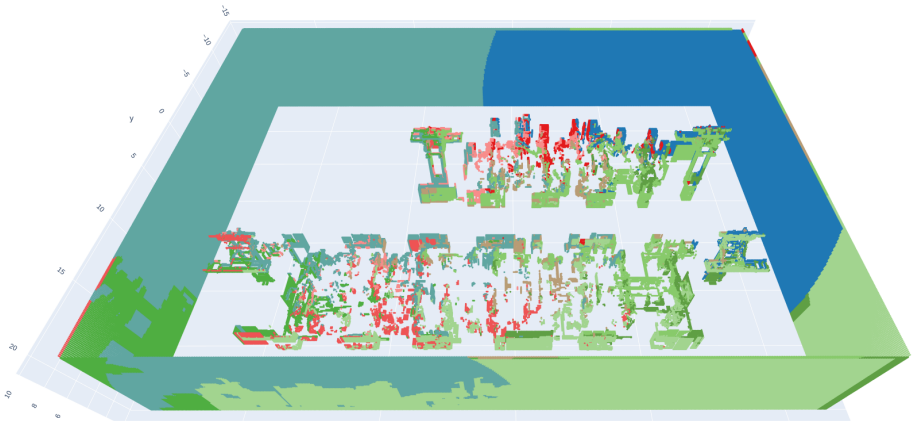


Figure 3.4: 3D visibility estimation from several SPs. Color represents individual SPs, i.e., points with the same color are seen from the same SP.

3.3.4 Scanning positions selection

The key part of the PK phase is the best SPs selection. The choice is based on the previously computed set of visible or visited points. The first SP is either the starting position or the position uncovering the largest part of the model, i.e., the one with the highest number of visible or visited points. This decision depends on the algorithm settings. A final visibility set is created as a union of the final visibility set and the set of visible points of the newly added SP. It is initialized to the set of visible points of the first SP. Then the SPs are chosen from SPCs based on the size of their contribution in the final set from highest to lowest up to a given maximum number of SPs or once the contribution to the final set is smaller than a given threshold.

3.3.5 Ordering of the SPs

Once the SPs are chosen, their order has to be determined. To do that, it is necessary to find the (shortest) paths between individual poses.

At first, the graph was constructed from the empty cells of the 2D prior model grid with appropriate edge weights ($\sqrt{2}$ for diagonal moves and 1 for other moves). Then Dijkstra's algorithm [30] was used to connect the SPs. The shortests paths were computed to obtain the distances (costs) between the SPs to obtain their order.

The SPs and calculated distances construct a complete graph. The minimum cost Hamiltonian path is computed to obtain the order of the SPs. The Hamiltonian path is the path that visits every vertex in the graph exactly once. Since the graph is complete, there is a path between every two vertices. Opposed to the traveling salesman problem, i.e., finding the minimum cost Hamiltonian cycle, there is no need to return to the start vertex. Nevertheless, both problems can be approximately solved using the same greedy technique.

The greedy algorithm holds the list of visited SPs and the existing route. The algorithm starts from the (selected) starting SP. The next SP is the one with the lowest cost from the current SP that was not already visited. Once the last SP is processed, the order of the SPs is generated and the PK phase ends. Example of the chosen SPs and their order is visualized in Fig. 3.5 together with 2D visibility estimation.

3.4 Next-Best-Scan phase

The NBS phase assures the environment will be covered sufficiently to detect the deviations in the object positions and orientations compared to the factory model. It starts with an initialization step when the initial scans are done to obtain initial knowledge about the environment (see Sec. 3.4.1). The point clouds from the scans are processed (see Sec. 3.4.2), then stored in the dynamic model and used for static model updates (see Sec. 3.4.3). The SPCs are chosen from the updated static model (see Sec. 3.4.4). For each SPC, its gain value is computed (see Sec. 3.4.5), and the SPC with the highest gain value is selected as the next SP (see Sec. 3.4.6). Then the process is repeated until the best SPC gain is too low or the time is exceeded.

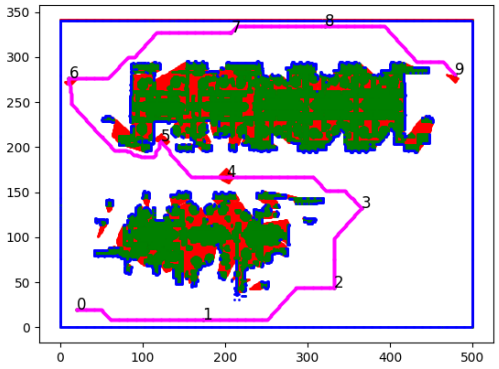


Figure 3.5: Visualization of the PK phase output. Numbers represent SPs and their order. A path between SPs is in magenta. The scanned empty space is in white, the unseen empty space in red, the scanned occupied cells in blue, and the unseen occupied cells in green.

3.4.1 Initialization

During the NBS phase initialization, we distinguish two different cases. In the first case, the prior model is available, the PK phase has been done, and the initial SPs are provided and ordered. The robot goes through the SPs and collect point clouds from static scans. In the second case, there is no prior information, and hence the scan in the robot starting position has to be made to receive the initial information about the world.

3.4.2 Point cloud gathering

While preparing the ESS algorithm, we assume the robotic platform carries two different scanners, one with lower angular resolution and the ability to scan the environment while moving (i.e., the LiDAR) and one with higher angular resolution without that ability (i.e., the terrestrial scanner). Example of collected data during an experiment can be seen in Fig. 3.6. Data from a LiDAR are used for robot navigation in the environment and dynamic model. Gathered data from the terrestrial scanner are the main output of our pipeline.

3.4.3 Model update

The dynamic model is updated directly from measured data, but the static model update is estimated from the dynamic model. Example of both models can be seen in Fig. 3.7.

Dynamic model update

The gathered point cloud from LiDAR is transformed from the scanner frame to the robot base frame. The transformed point cloud is inserted into the dynamic model. For each point of the point cloud, the corresponding voxel is found and marked as occupied.

Two post-processing procedures were implemented to filter unwanted fragments in the point cloud. The first one is a simple (but not robust) eraser of dynamic objects. This methods resolves the problem caused by people walking around the robot. Then the algorithm could

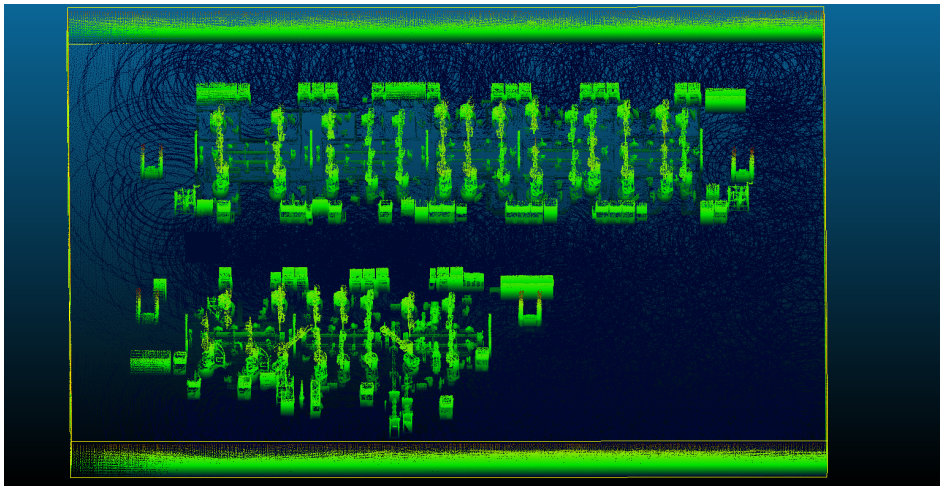


Figure 3.6: Gathered point clouds during an experiment in simulation, colored to distinguish floor from factory equipment.

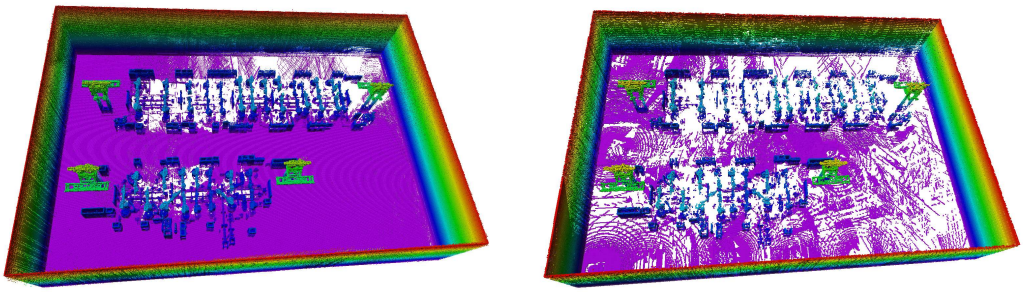


Figure 3.7: World model constructed from LiDAR data (left) and from terrestrial scanner data (right) after several static scans represented in the octree structure.

not find any SPC because the point clouds contained the points representing the human body. The eraser assumes that the points of the model which are situated in front of the currently gathered points cannot be in the model; otherwise, the currently gathered points would not be collected. Therefore, rays are cast from the robot's position to the currently gathered points while processing a new measurement. If any point in the dynamic model lies in the ray path, it is removed from the dynamic model. Due to its time complexity, the eraser is called only in every tenth point cloud processing.

The second procedure is removing a human operator who is walking with the robot and controlling the scanning process. The operator's position is restricted to be behind the robot (for simplification). As the human operator follows the robot and corrupts a part of the field of view, removing these corrupted points from the measurements is necessary. Therefore, the measurements from this circular sector (with a central angle $\theta = 60°$), which are closer than 2 m to the scanner, are skipped and not added to the dynamic model.

Approximated static model update

As the gathered high-resolution point clouds from static scans are not available during the experiment, these collected data must be approximated. The approximation exploits the information about the environment gathered by the LiDAR while the robot is moving, i.e., dynamic model.

The estimation process is similar to the special case of the 3D visibility computation. All voxels of the world model within a given maximum distance are tested whether they are visible from the SP. The visible ones are added to the world model. Then the rays are cast from the SP to these points. The traversed voxels are marked as empty because we assume the points would not be collected if any of them was occupied. The estimation depends on the quality of the dynamic model. Therefore, the first approximation is done after processing of all SPs prepared in the PK phase and for all of them at once. The reason is that after going through these SPs, the environment should be reasonably covered, which increases the static scan estimation accuracy. However, this method still underestimates the world coverage leading to redundant SPs. After the first approximation, the static model is updated only for a new SP, not all previous ones.

3.4.4 Scanning position candidates sampling

The current position neighborhood is sampled in 2D space using the subsampling method with a given step, as in the PK phase. The step value depends on the neighborhood size to have always almost the same number of SPCs. The size of the neighborhood is a parameter that should be tuned for the specific environment. Each sample is then disturbed by adding a normally distributed noise ($\mathcal{N}(0, 1)$) to both coordinates.

The disturbed sample can become a SPC if it fulfills two conditions — the distance between the current position and the new one must be higher than a specified threshold, and the new position and its neighborhood representing the robot's body must be known in the world model and not occupied. As in the PK phase, if the disturbed position does not satisfy these conditions, it is disturbed and checked again (up to five times). If the number of found SPCs is less than a specified threshold, the neighborhood size is increased and the process of their selection is restarted.

3.4.5 Gain computation

The gain computation takes into account the necessity to cover the environment properly to gather enough data needed for the factory installation check. Therefore, the gain represents how much of the yet unknown space could be explored from given SPC, and is estimated using the ray-casting method. The rays are cast in the world model from the SP in the given directions. If the ray intersects the first unknown voxel, it is assumed that it would explore it and increase the knowledge about the environment. The angular resolution was estimated by computing the required angular resolution to cover all voxels with given resolution at a specific distance from the SP because the computational time is limited. Based on analysis of scanned data, the mean distance from the measured points to the SP is around 6 m for the factory model. The voxel edge length is assumed to be 0.1 m, and hence the angular

resolution α can be computed as

$$\tan\alpha = \frac{0.1}{6} \quad \rightarrow \quad \alpha = \arctan\frac{0.1}{6} \approx 0.95^\circ \approx 1^\circ. \tag{3.6}$$

This change motivates the robot to come closer to the objects.The intersected unknown voxels are counted only if they are in a given height interval to eliminate the influence of the floor and roof on the gain value, i.e., the manipulators and other factory equipment are prioritized over the building itself. Moreover, the distance between the unknown voxel and the SP must be higher than the minimum scanner range. The gain value is the number of these intersected unknown voxels.

3.4.6 Next scanning position selection

The next SP is the SPC with the highest computed gain value. Once the gain of the selected SP is lower than a threshold, the robot returns to the starting position. Then the experiment either stops or continues from the starting position. This return to the start may help to reach other parts of the environment which are too far from the current position.

3.4.7 Model division into parts

During the ESS algorithm preparation, the idea of dividing the large environment into several parts and processing them sequentially was considered. Therefore, the ESS algorithm supports the possibility to run the PK phase one part after another with a specific number of SPs for each part. Then these SPs can be used in two different ways in the NBS phase.

1. The model division is not used in the NBS phase, and the prepared SPs from the PK phase are taken in their order computed as usually, but the path may not be the same one as if the model was not divided. The NBS phase starts in the last part, and there is no area restriction for SPCs.

2. The model division is used in the NBS phase as well, and the parts are processed one after another. It means that the next SP selection is limited to the area of the part.

3.5 High-level robot controller

ESS algorithm can be used without the robot, as it was used for preliminary tests of the ESS algorithm (see Sec. 4.2). In that case, it is necessary to move the scanner between the SPs manually. Another option is to control the robot movements manually, i.e., via teleoperation. Nevertheless, the main task of the thesis was to prepare an automated scanning process, including the autonomously moving robot. Therefore, it was necessary to prepare a high-level controller which communicates with the RDS packages providing autonomous robot control (see Sec. 2.5) on the lower level. The robot high-level behavior is controlled by a state machine.

3.5.1 Robot behavior state machine

A diagram of the state machine is pictured in Fig. 3.8 with individual states and transitions between them. The states are listed as follows:

- IDLE — The controller stays in IDLE state during the startup procedure and when the process is paused.
- SCANNING — The high-resolution scan is being gathered. In real-world scenarios, this procedure takes several minutes. The next SP is computed during the time needed to complete the scan.
- READY TO MOVE — Robot is ready to move, the target position is sent to the navigation program, and the confirmation is awaited.
- MOVING — Robot is moving towards the target position.
- STUCK — Robot is stuck. The command to stop the robot's movement is sent.
- STOPPED — Robot is stopped. As the last several trajectory positions were saved, the first safe checkpoint is selected as the next target.
- GOING BACK — Robot backtracks to the selected checkpoint.
- TELEOP — Robot is controlled externally via a teleoperation controller.
- END OF EXPERIMENT — Robot is returning to the starting position, and the program is paused.

At the beginning of the experiment, the controller starts in IDLE state. Once everything is prepared, the state is changed to SCANNING state if the starting position is SP, otherwise it is skipped. Once the scan is finished, a new target is sent to the navigation program, and the state is changed to READY TO MOVE. When the target confirmation is received, the state is changed to MOVING, and the robot starts moving.

If the robot cannot move towards the target (this information is received from the RDS), the state is set to STUCK. Otherwise, the target is reached or the reaching timeout is exceeded, and the state is changed to SCANNING. Once the robot movement is stopped via the sent command in STUCK state, the state is changed to STOPPED, which remains until the checkpoint target is confirmed, then the state is GOING BACK. The state is again changed to SCANNING when the checkpoint is reached or the time runs out. Sometimes it is necessary to take control of the robot's movement using the teleoperation controller. In that case, once the teleoperation controller is activated, the state is changed to TELEOP. When the robot control is autonomous again, the state is changed back to the previous state.

Once the experiment timeout is exceeded or the maximum number of SPs is reached, the starting position is set as the new target and the state is changed to END OF EXPERIMENT. Then after reaching the starting position, the state is changed to IDLE. If the program starts again, the state is changed to SCANNING.

3.5.2 Teleoperation controller

A game controller (button layout can be seen in Fig. 3.9) serves as the teleoperation controller of the robot. Its primary feature is controlling the robot's movement, and it has a higher priority than the autonomous robot control. It means that once the game controller takes over the robot control, the velocity commands from the RDS packages have no impact. The robot

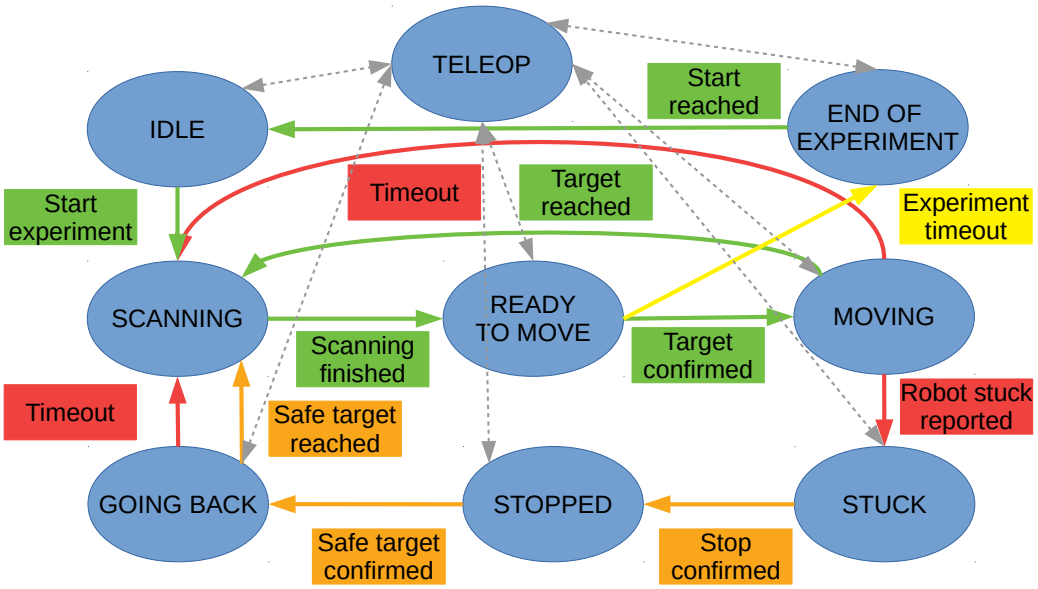


Figure 3.8: High-level robot controller state machine. States are in blue ellipses, and the rectangles represent the transition conditions. Optimal transitions are green, red represents the unwanted transitions, orange represents transitions to recovery from unwanted states, and gray represents transitions to and from the TELEOP state.

is moved with the left joystick (LSB) while holding the left or right bumper (LB/RB). Once the bumper is released, the robot stops and eventually takes commands from the autonomous control programs.

Compared to the controller usage during DARPA SubTerranean Challenge [17] described above, several other buttons were mapped to specific triggers to accelerate the experiments and make them safer for the robot and the environment. One of these buttons resolves the issue that there is currently no possibility to automate the static scan process. There is no communication between the program and the terrestrial scanner, which means that the scanning procedure must be started manually, and then the program cannot detect that the scan is finished. The newly mapped buttons are listed as follows:

- Right Trigger (RT) — The other pressed buttons are registered only when this button is held. It serves as a safety button against accidentally pressed buttons.
- Button A — Pressing this button signals that the high-resolution scan is finished, and the robot can move again.
- Button B — Pressing this button triggers resuming of the previously paused experiment.
- Button X — Pressing this button sends the robot to the starting position and pauses the experiment.
- Button Y — Pressing this button interrupts the robot movement, and the current position is marked as the target SP. This button is usually used to skip waiting for a timeout when the robot is stuck or the position is unreachable.

- Start Button — Pressing this button sends a command to the RDS navigation program to clear its local map. This command can be helpful for elimination of unexpected noise or dynamic objects in the robot's neighborhood.
- Left Trigger (LT) — The stop command is sent to the robot once this button is pressed. When this button is held, the robot is in the teleoperation state. Once the button is released, the robot starts moving autonomously again.

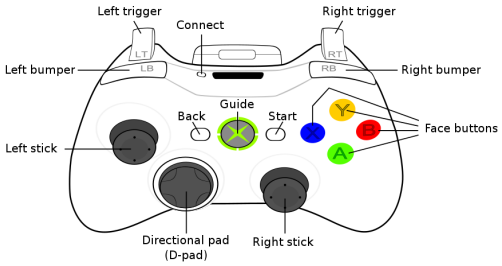


Figure 3.9: Xbox 360 wireless controller[40].

3.6 Evaluation methods

We prepared several evaluation methods. The first of them is used for comparison of variants of the proposed algorithm. The rest of the metrics are used to evaluate the measured data.

3.6.1 Environment coverage metric

The proposed exploration algorithm exists in different variants. The selection of the best ones is decided based on the variant's ability to cover as much of the whole environment as possible. The explored area represented as occupied and free nodes in the octree model was compared with the ground-truth model in the form of a voxel grid with occupied and free voxels. For each voxel in the ground-truth model, we check whether it is explored. i.e., stored in the octree model, and then if it is occupied or not (see Tab. 3.1 for notation).

notation		**original model** free (G_f)	occupied (G_o)
measured model	free	H_f	M_o
	occupied	M_f	H_o
	unknown	U_f	U_o

Table 3.1: Voxel comparison notation.

The similarity score S representing the similarity between the ground-truth model and the explored area after the experiment is computed as:

$$S = \frac{H_o + 0.01H_f - M_f - M_o}{G_o + 0.01G_f}, \tag{3.7}$$

where the correctly marked occupied space (H_o) has the highest weight (equal to 1) because it corresponds to correctly gathered points in the point cloud. The correctly marked free space

(H_f) has a significantly lower positive weight because of the high ratio of free to occupied voxels, and the free space is not gathered but only assumed. The unknown (not measured) space (U_f and U_o) does not occur in the equation because it does not make the model better or worse, so its weight equals zero. On the other hand, the wrongly marked space (free if it is actually occupied M_o or the other way M_f) makes the model worse, and thus it has a negative weight. The similarity score ranges from -100 to 1, but values under zero should not be usually seen, and higher values signify higher similarity between the compared models.

Sometimes it is undesirable to mark the free space in the OctoMap tree due to high memory load. In that case, the modified similarity score is computed as:

$$S_M = \frac{H_o - M_f}{G_o}, \tag{3.8}$$

score ranges from -1 to 1, and higher values signify higher similarity between the compared models.

3.6.2 Metrics for algorithm performance evaluation

Once the best algorithm variants are chosen, it is necessary to test them in the planned application with measurement imperfections (e.g., scanner noise, inaccurate robot target position reaching). Several different metrics were prepared to evaluate the algorithm performance.

Robot localization error

First of all, the robot localization error should be evaluated. The precise localization is crucial for the task from several points of view. The robot has to know where it is to reach the given target position and safely avoid all obstacles along the path. Another important reason is the necessity of a meaningful scanner pose estimation to increase registration speed and precision. The registration algorithm in Leica BLK360 finds overlapping data in individual point clouds and matches them when no additional markers are used.

Unfortunately, the localization error can be evaluated only in the simulation because there is no ground truth robot position in real-world scenarios. Nevertheless, we can detect the problematic parts of the given model, where the error increases, and possibly solve them in the real world by adding additional features like, e.g., markers in the specified positions.

Gathered point cloud density

This metric evaluates the proportion of the model (density), which is covered by the collected point cloud. The distance from each point in the reference model to the nearest point in the point cloud is computed. Optimally, each point should have zero distance from the corresponding point in the point cloud. Distances higher than zero can be caused probably by one of the two reasons — 1) The corresponding point was not gathered, and thus the distance is computed to a different one; 2) The corresponding point was collected in a different location than assumed due to measurement noise or object displacement. The metric output can be visualized, e.g., in form of a heatmap computed in CloudCompare program [41] (example in Fig. 3.10).

The output heatmap shows the measured data distribution in the world. It can be helpful to identify the missed objects or their parts. However, this metric does not contain semantics. Therefore, if the collected point cloud was monolithic and infinitely dense (with almost zero distance between neighboring points), then the distance between the model points and their correspondences in the point cloud would be almost zero, but no object could be recognized.

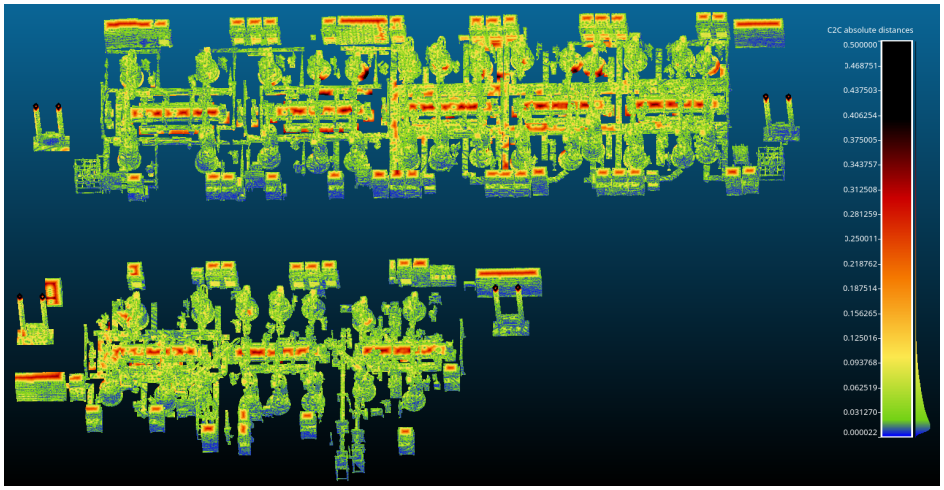


Figure 3.10: Metric visualization example—gathered point cloud density heatmap example.

Estimated 6D pose error of the objects

Estimated 6D pose error of the objects is an evaluation method that simulates a check of the position and orientation of individual robot manipulators and robotic cells. This metric computes the difference between the object 6D pose in the reference model and its assumed 6D pose in the gathered point cloud. Individual objects from the reference model are aligned using the ICP algorithm (see Sec. 2.4.1) with the collected point cloud. The resulting transformation between the object pose in the reference model and the point cloud is either an error caused by insufficient data or the object displacement detection. The example of the metric visualization can be seen in Fig. 3.11.

The alignment procedure may struggle with similar models lying next to each other, e.g., several closets in a row. Once the point cloud is too sparse, an object may be placed at the position of a similar one or between the two of them. Moreover, it is sensitive to registration errors and scanner pose estimation errors.

The estimated pose error computed here is just a preview of the results because the actual check of the position and orientation of individual robot manipulators and robotic cells is evaluated in a specialized program in practice.

Detection of shifted objects

Once the poses of the objects are estimated, it can be decided whether they are in their predetermined position or not. As we assume the 2D problem only (the floor is not sloped), the position check is done only in 2D and without orientation error for simplification. This leads to a binary classification task, where class 0 means the object is in its predetermined

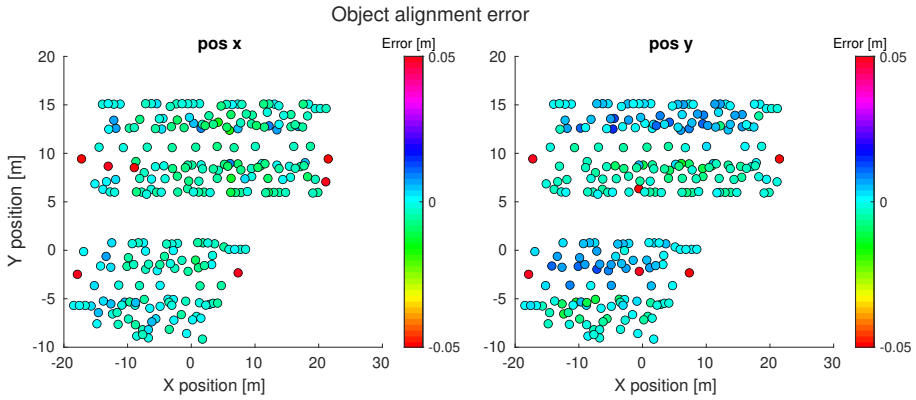


Figure 3.11: Metric visualization examples—estimated pose error of the objects in individual coordinates.

position in the x-y plane, and class 1 (i.e., positivity) means it is shifted in the x-y plane. For binary classification, four different combinations of classes are defined (see Tab. 3.2) and written in a form of a confusion matrix C.

binary classification		**Assigned class**	
		0 (not shifted)	**1 (shifted)**
Actual class	**0 (not shifted)**	True Negative (TN)	False Positive (FP)
	1 (shifted)	False Negative (FN)	True Positive (TP)

Table 3.2: Binary classification combinations.

In this thesis, we use two metrics to measure the performance of our classifier (object pose estimation). The first one is accuracy, the ratio of correct predictions out of the total number of cases tested. The second one is sensitivity, the ratio of correctly classified positive cases (i.e., shifted objects) out of all classified positive cases.

$$C = \begin{bmatrix} TN & FP \\ FN & TP \end{bmatrix} \tag{3.9}$$

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{3.10}$$

$$\text{Sensitivity} = \frac{TP}{TP + FN} \tag{3.11}$$

Chapter 4

Experiments and Results

In this chapter, we present the experiments and their evaluation for the factory mapping process. At first, we tested the frontier-based exploration algorithm (further referred to as Naex experiments) in Sec. 4.1. Then the optimal variants of the proposed algorithm were searched in Sec. 4.2. We performed experiments for the optimal variants of the Exploration by Static Scans (ESS) algorithm in the robotic simulator (see Sec. 4.3) and in the real-world scenario (see Sec. 4.4).

In this thesis, an experiment refers to a factory mapping process, in which the robot autonomously traverses the environment for a specific amount of time and collects scans of the factory equipment, and when the timeout is exceeded, the robot goes back to the starting location. The model used in all experiments except for the real-world experiment is a model of a factory hall.It consists of several robotic cells separated by fences and accessible through doors. Each cell contains multiple industrial manipulators, other tools for car assembly, and cable channels distributing electricity. In this thesis, the (re-)construction phase without fences and doors is assumed to eliminate the problem of door detection and increase the probability of scanning parts of the objects otherwise invisible (occluded) due to cell separation. The world coordinate system can be seen, e.g., in Fig. 4.3, with the z-axis going up. The model was prepared with the model preprocessing pipeline (see Appendix A).

The real-world scenario took place in Testbed for Industry 4.0 at Czech Institute of Informatics, Robotics and Cybernetics, Czech Technical University in Prague [42](further referred to as Testbed). Unfortunately, the prior model for this environment does not correspond to the current state. Therefore, no ground truth model for this environment was available.

4.1 Frontier-based exploration algorithm evaluation

Two simulation experiments were performed to evaluate Naex (see Sec. 2.6) and to test the prepared metrics. The reference model was used in the first experiment (Naex 1). For the second experiment (Naex 2), the model was modified — several objects were moved to different positions. This imitates the real-world situation where it is necessary to detect the deviated objects. The collected point clouds were stored in the robot's built global maps. Two global maps were constructed for these experiments compared to the usual usage of only one map. The first one was the robot's main map used for its movement planning with distances between neighborhood points around 15 cm, and the second one was an additional map with a finer resolution (the distances between points were around 2 cm). The gathered point clouds were retrieved from these two global maps.

4.1.1 Naex experiment in the reference model

The first experiment lasted 35 minutes, and as can be seen in Fig. 4.1, the robot traversed the environment several times. At first, a gathered point cloud density was computed for the collected output data. The density is visualized as a heatmap in Fig. 4.2. As shown, the bottom part of manipulators and the bottom part of closets are covered densely. On the other hand, all objects are insufficiently covered from the top view, which is not visible from the mobile robot.

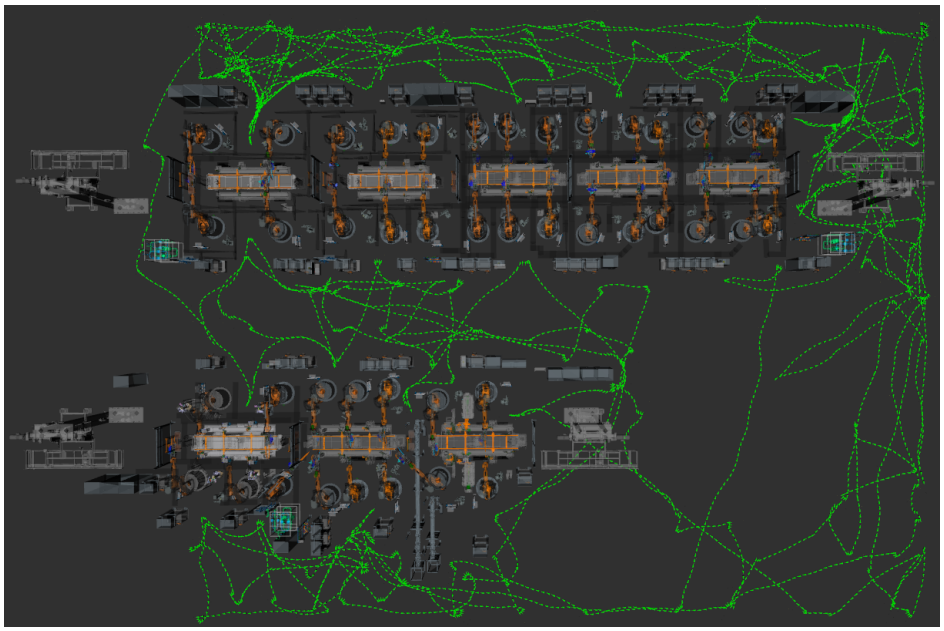


Figure 4.1: Robot trajectory during a Naex 1 experiment.

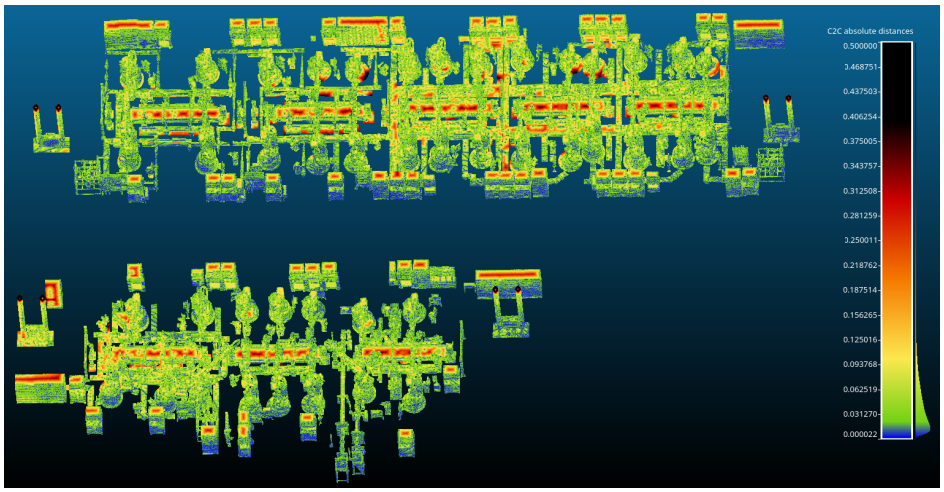


Figure 4.2: Point cloud density as heatmap of distances between reference and measured points for the Naex 1 experiment.

The next step was calculating the estimated pose error for all 250 objects. As the reference model was used in the experiment, the estimated pose error should always be zero. Thus,

nonzero values are caused by noise or insufficiently scanned objects. In this experiment, the entire world model seems to be covered. The computed estimated pose errors are visualized in two ways. Figure 4.3 shows individual objects represented by colored circles in the x-y plane, representing the environment 2D projection. The color encodes the size of the pose error in the given coordinate. As shown, the highest position deviations are in the z-axis. Furthermore, it can be noticed that there is a correspondence between the position errors in the y-axis (left center) and the z-axis (left bottom), e.g., a horizontal stripe for y coordinates around 12 m. Objects in this stripe are mainly the manipulators.

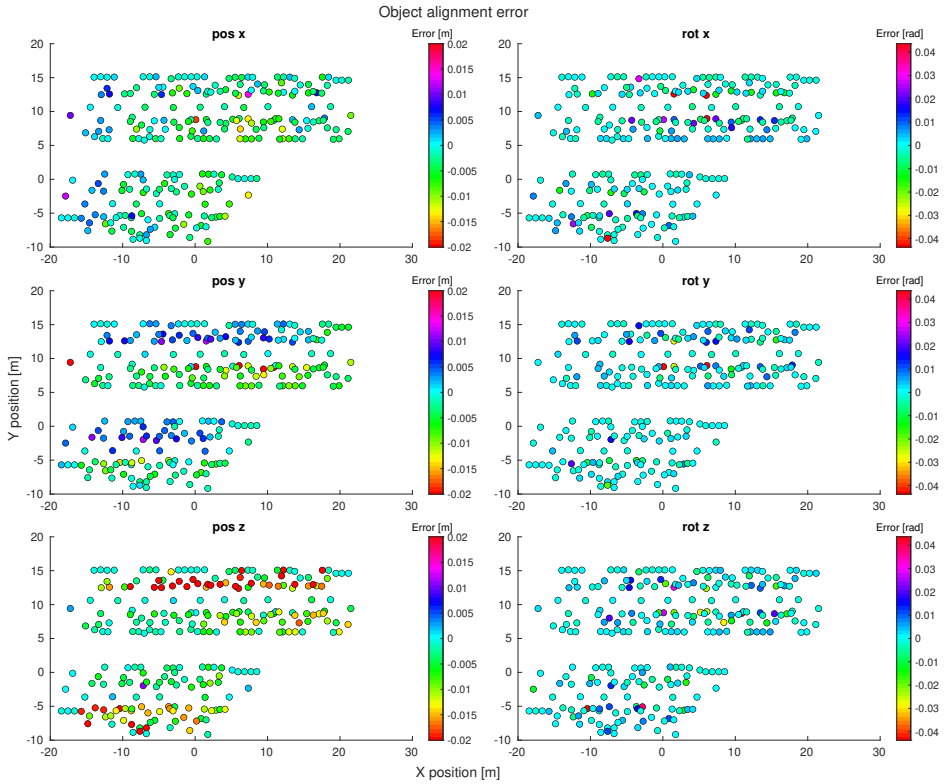


Figure 4.3: Object positions and deviations for the Naex 1 experiment. Circles represent individual objects.

Figure 4.4 shows the same estimated pose errors this time as the scatter plot with border lines for ±0.02 m (red line) and ±0.01 m (green line). As can be seen, all objects are within ±0.02 m error in the x-axis, and only four objects are out of the ±0.02 m interval with the error in the y-axis. Most objects have errors within ±0.01 m for x and y coordinates. However, almost a third of the objects have an error around or above ±0.02 m in the z-axis.

The localization error, shown in Fig. 4.5, is the most significant in the z-axis, with a maximum value of about 0.03 m. It seems that the localization error causes the object alignment error if we put these two errors into context. The localization error in the x and y axes is hardly ever out of the ±0.01 m interval.

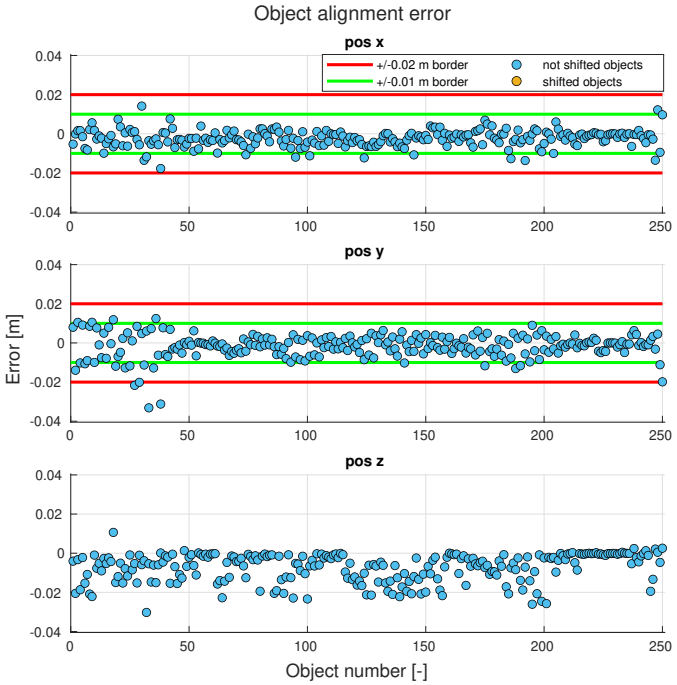


Figure 4.4: Object positions deviation as a scatter plot for the Naex 1 experiment.

Figure 4.5: Robot estimated position error during the Naex 1 experiment.

4.1.2 Naex experiment in the model with 9 shifted objects

As opposed to the previous experiment, nine objects (three manipulators, four lifters, and two additional objects) were selected, and their positions were changed by values from ±10 up to ±20 cm in the x-axis, y-axis, or both of them. This experiment aims to test whether the object pose error is computed correctly to detect the deviations and align the objects properly.

The experiment lasted longer than the previous one; however, the number of points in the maps was almost the same. The reason is that the robot traversed the environment repeatedly without visiting new places.

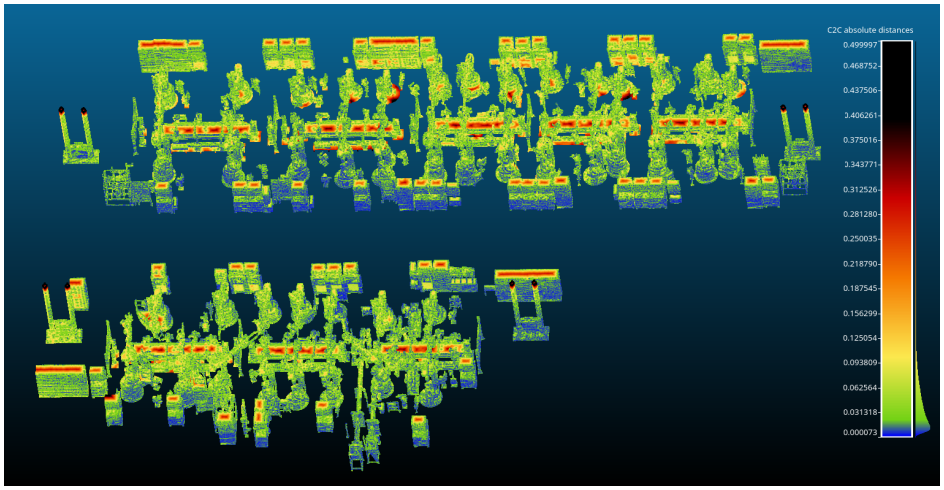


Figure 4.6: Point cloud density as heatmap of distances between reference and measured points for the Naex 2 experiment.

Since the reference model is not the same as the test one, the point cloud density has to be computed using the test model (see Fig. 4.6). Otherwise, the metric would not work correctly because the shifted objects would seem to be insufficiently covered.

Figure 4.7 shows object positions projected into the x-y plane. Colored squares represent the objects which were shifted and colored circles the rest. As can be seen, only the squares are red signalizing that their position errors are outside the interval ±0.02 m, except for the case of z-axis position error.

Orange circles in Fig. 4.8 represent the shifted objects. The figure shows the pose errors computed in the test model (even shifted models should have zero error now). It can be seen that the pose errors of the shifted objects are inside the ±0.01 m interval.

The localization error (see Fig. 4.9) is similar to the one in the previous experiment. Again, the z-axis error is within 0.03 m in most cases, and the error in the x and y axes is hardly ever out of the ±0.01 m interval.

4.1.3 Classification of shifted objects

The promising results after alignment were used to decide whether the objects are in their predetermined position or not. The detection of shifted objects was done only in x-y plane

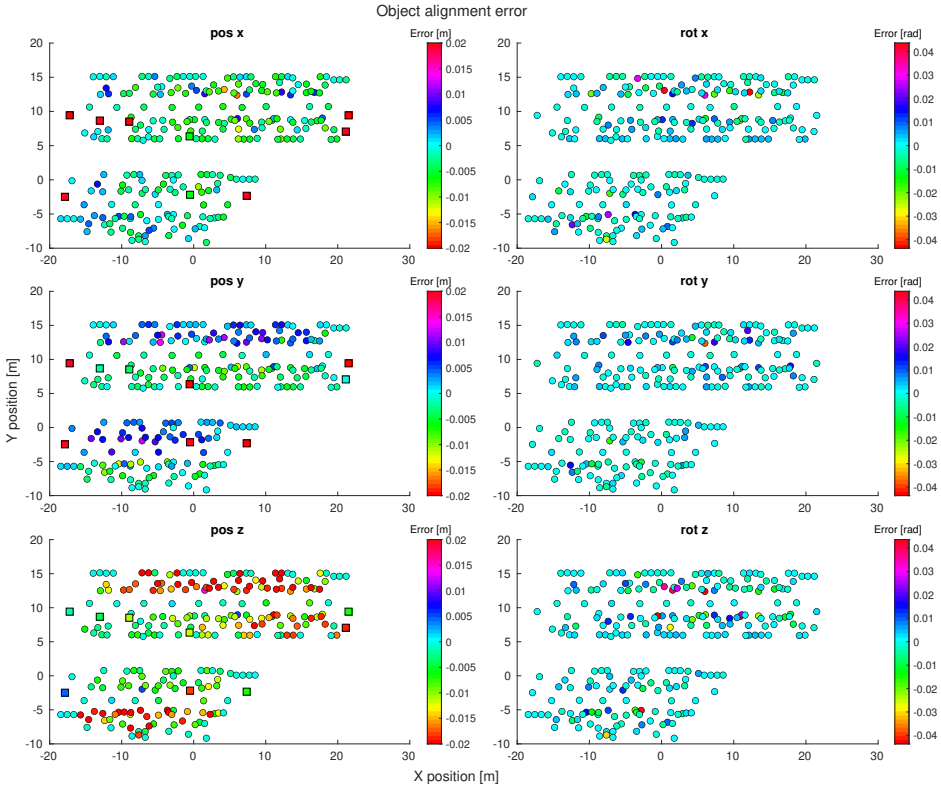


Figure 4.7: Object positions and deviation for the Naex 2 experiment. Squares represent shifted objects and circles the rest.

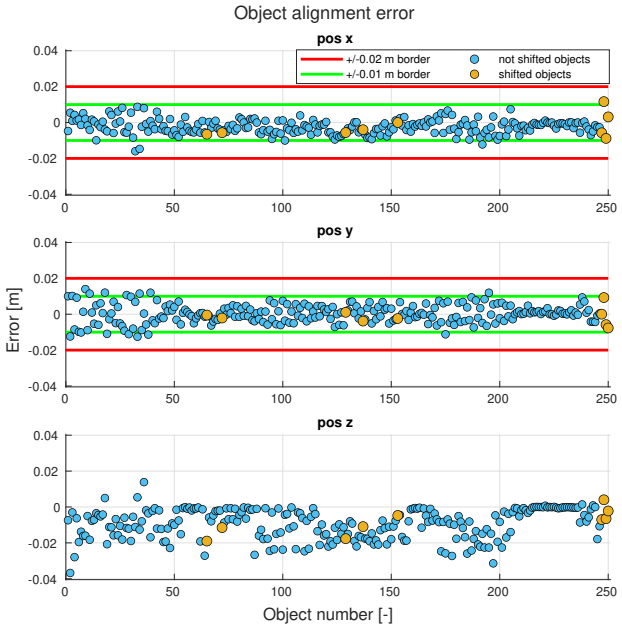


Figure 4.8: Object positions deviation as a scatter plot for the Naex 2 experiment.

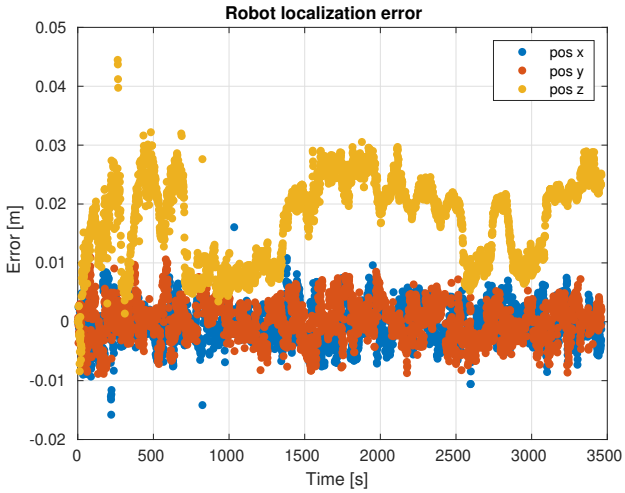


Figure 4.9: Robot estimated position error during the Naex 2 experiment.

without rotation. A decision whether the object was shifted was made regarding the threshold, which was set to the same value for both coordinates. The confusion matrices (see Tab. 4.1) and classification accuracy (see Tab. 4.2) were computed for four different values of the threshold — 0.005 m, 0.01 m, 0.02 m, 0.05 m.

As there were no shifted objects in the Naex 1 experiment, all objects should be ideally predicted as not shifted. On the other hand, there were nine shifted objects in the Naex 2 experiment, and the offset value for a specific coordinate is higher than or equal to 0.1 m. As can be seen in Tab. 4.1, there are no shifted objects marked as not shifted for all threshold values in both experiments. Moreover, all shifted objects are marked as shifted for all threshold values in both experiments. The only problem is marking not shifted objects as shifted (i.e., false positive — FP). In that case, the threshold value significantly impacts results, as shown in Tab. 4.2, where the accuracy increases (up to 1) with a higher threshold value.

Experiment	**threshold**							
	0.005 m		**0.01 m**		**0.02 m**		**0.05 m**	
Naex 1	139	111	216	34	246	4	250	0
	0	0	0	0	0	0	0	0
Naex 2	119	122	222	19	241	0	241	0
	0	9	0	9	0	9	0	9

Table 4.1: Confusion matrices for both Naex experiments. Matrices are written as $\begin{matrix} TN & FP \\ FN & TP \end{matrix}$, where positivity means that objects are shifted.

Experiment	**threshold**			
	0.005 m	**0.01 m**	**0.02 m**	**0.05 m**
Naex 1	0.556	0.864	0.984	1
Naex 2	0.512	0.924	1	1

Table 4.2: Accuracy of shifted objects detection.

4.2 Evaluation of ESS algorithm variants

Since several aspects were not decided during the implementation of the ESS algorithm, all their possible variants were implemented. These aspects can be specific for different environment types. Therefore, the preliminary tests of the ESS algorithm, mainly its different variants and settings, were made outside of the robotic simulator. The reason was to eliminate the influence of the robot movement constraints, target reachability, movement uncertainty, and scanner uncertainty, so the results were affected only by the actual algorithm settings. Therefore, we assume the scanner is a 3D sphere robot (nearly a point robot) with a precise placement accuracy and all objects have an ideal reflectivity at an arbitrary angle. Moreover, we assume the scans are perfectly aligned.

4.2.1 Evaluation program and model environments

The prepared testing program is based on OctoMap library, allowing us to cast rays in the discretized environment easily. The program can take a list of Scanning positions (SPs) prepared based on the prior model knowledge as an input. The scanning simulation is done by ray-casting (provided by OctoMap) from the SP towards the known model of the environment. Once the ray reaches the occupied space, the voxel center representing that space is stored in a simulated point cloud, and the ray does not continue further. Since two different scanners on the robotic platform are assumed, two different scanning simulations are done to see the difference between scanners. Once the test finishes, the similarity score for both scanners is computed and compared against other ESS algorithm variants.

Two different models (see Fig. 4.10) were provided for these experiments — 1) a **factory** model with dimensions around 50x34x10 m with a resolution of 10 cm; 2) a **cell** model with dimensions around 15x12x10 m with a resolution of 2 cm. The second model simulates independent scanning of individual robotic cells. Three disturbed versions (see Tab. 4.3 for more details) exist for each model simulating the real state with wrongly placed objects. These versions served to test the influence of prior model accuracy. Moreover, the exploration performance is tested for two different scanner locations and different combinations of angular resolution for both scanners. All testing cases can be found in Tab. 4.4.

Model version	**ratio of shifted objects**	**disturbance distribution**
small differences	1/3	$\mathcal{N}(0, 0.01)$
medium differences	2/3	$\mathcal{N}(0, 0.25)$
big differences	1	$\mathcal{N}(0, 1)$

Table 4.3: Model versions overview.

4.2.2 PK phase

At first, the Prior Knowledge (PK) phase variants (see Tab. 4.5) were tested to find the most effective ones for the factory environment. The variants were tested in different scenarios and settings to assess their performance. Since a pseudo-random generator is used for the generation of the Scanning position candidates (SPCs), every experiment was repeated three times with different seeds.

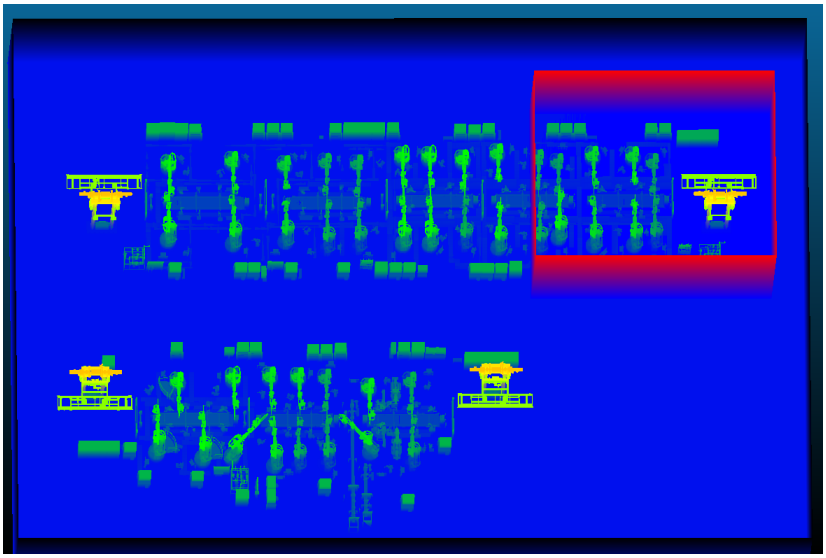


Figure 4.10: Whole factory model with the selected cell model.

Parameters		**Values**		
Angular res. (hor./ver.)	ter. scanner	0.1° / 0.1°	0.1° / 0.1°	0.2° / 0.7°
	LiDAR	0.2° / 0.7°	1° / 1°	1° / 1°
	label	(Ang comb 0)	(Ang comb 1)	(Ang comb 2)
Sensor z coordinate		0.8 m	1 m	
Model type		factory	cell	
Model disturbance		Small	Middle	Big

Table 4.4: Experiment settings combinations.

Since the variants were tested for different sensor z coordinates (height) and different model versions, the influence of these settings on the similarity score S_M (see Sec. 3.6.1) was checked. As shown in Fig. 4.11, the similarity score is almost the same for both sensor z coordinate values and all three model versions. It means the sensor placement on the scanning platform does not affect the results; hence, the sensor placement in the hardware prototype was not further optimized. Similarly, the model disturbance does not affect the results. It can imply that a higher number of SPs computed in PK phase could be preferred over a higher number of SPs computed in Next-Best-Scan (NBS) phase.

The actual comparison of individual ESS algorithm variants is shown in Fig. 4.12. Each row contains computed similarity score for different algorithm variants and experiment settings, with one algorithm strategy aspect taken as the compared variable. Results for the two methods of the first SP selection are shown in the first row. The results are the same for almost all variants. The only difference is when the model is divided into parts with only one SP used in each part (visualized in the bottom row as well). Otherwise, the inclusion of the starting position as a SP does not worsen the results. Hence, the starting position can be usually used, and its known position can be exploited for the robot's localization.

The prior model dimensionality is the compared variable in the middle row of Fig. 4.12. As can be seen, the similarity scores for the 3D prior model are higher in most cases. Therefore,

# of SPs	low - 1/8 (1 part/whole)						high - 5/15 (1 part/whole)					
Model division	None		8 parts				None		8 parts			
First SP	fixed		fixed		best		fixed		fixed		best	
# of dimensions	2	3	2	3	2	3	2	3	2	3	2	3

Table 4.5: All tested variants of the algorithm PK phase.

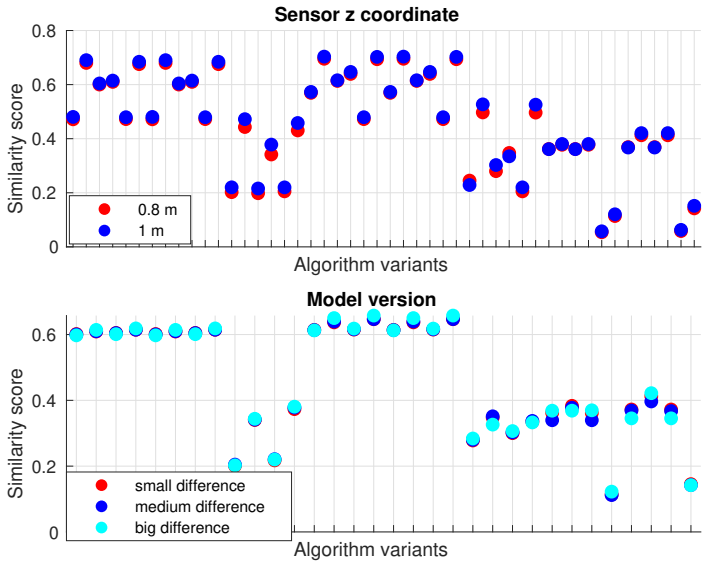


Figure 4.11: Similarity score for different sensor z coordinate (top) and different model disturbance (bottom).

the 3D prior model will be preferred for the PK phase if available.

The last row of Fig. 4.12 contains the model division strategy as the compared variable. In this case, the comparison is made only for the same total number of SPs prepared in PK phase to prevent influence of a different number of SPs. As shown, it should be preferred not to divide the model into separated parts but to process the whole model at once. The influence of the first SP selection can be seen here as well (as mentioned above).

4.2.3 NBS phase

The goal of the NBS phase experiments is to analyze an influence of SPs division ratio, i.e., how many of the SPs should be computed ahead in PK phase. Furthermore, the algorithm variant without PK phase is tested and then compared with the other ones.

As opposed to the previous experiments, the only used settings were the angular resolution and model type because the model disturbance and sensor z coordinate value did not affect the result in the offline phase. During the experiments, 30 scans were gathered, the sensor z coordinate was set to 1 m, and the chosen model version was the middle disturbed one.

Figure 4.13 shows a similarity score evolution during the experiments for the case without prior knowledge, the case with 8 SPs prepared in PK phase, and the case with 15 SPs prepared

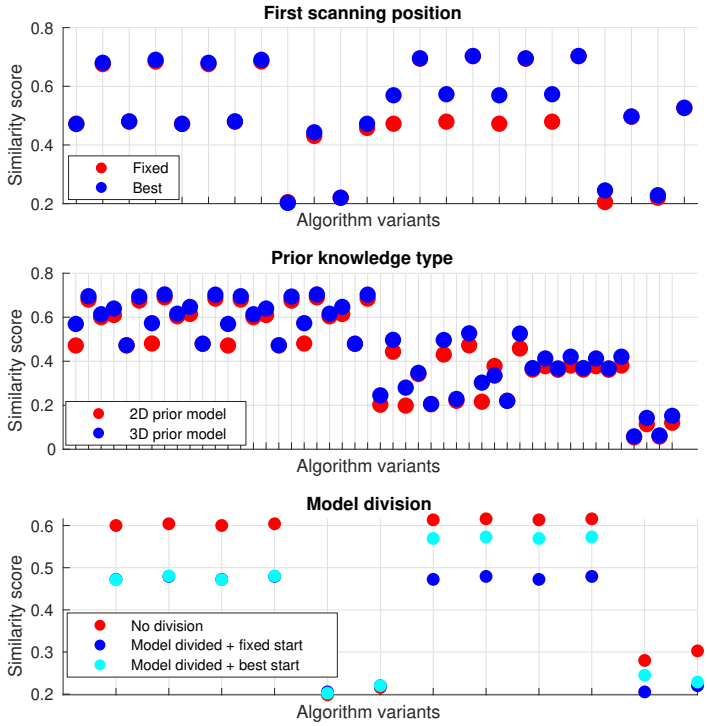


Figure 4.12: Similarity score for different strategies — first SP strategy (top), prior knowledge type (middle), model division strategy (bottom).

in PK phase. It can be noticed that prior knowledge about the environment expedites the model coverage. The similarity score after 30 scans for the case without PK phase is similar to the one after 6 scans for the case with 8 SPs in PK phase. Furthermore, the score for the case with 15 SPs from PK phase has slower growth, but the final score is higher than for the case with 8 SPs from PK phase. The slower increase of the similarity score during the PK phase can be caused by the order of the SPs — the SPs in the middle of the path have a lower impact on the score than the ones in the end. The final score comparison shows, that the model is more covered when a higher number of SPs computed in PK phase is used.

Moreover, the influence of the LiDAR angular resolution can be seen. The first case (Ang comb 0) has a higher similarity score for all three variants, and the most significant difference between LiDAR angular resolutions is when no prior knowledge is used, i.e., PK phase is skipped. The reason is that the LiDAR angular resolution affects the model estimation. Therefore, the higher LiDAR angular resolution estimates the already scanned environment more precisely, which leads to a more optimal next SP selection.

Finally, it can be seen that the similarity score becomes stagnant early in the NBS phase for the cases with PK phase. The reason is that the environment estimation by the LiDAR is not sufficient; thus, the selected SP do not significantly improve the environment model. This issue is addressed in Sec. 5.1. Nevertheless, the scanned world approximation was modified for the experiments in the robotic simulator and the real-world scenario.

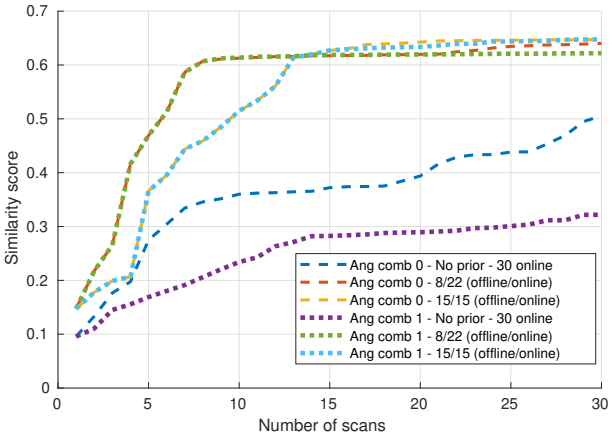


Figure 4.13: Evolution of similarity score for different PK phase strategies and different angular resolutions.

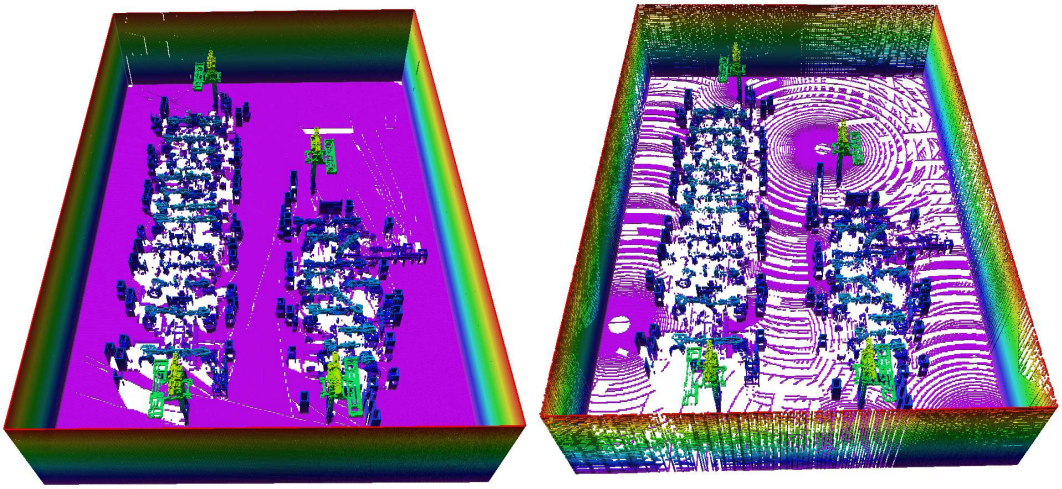


Figure 4.14: Model of the environment based on the terrestrial scanner data (left) and LiDAR data (right).

4.3 Algorithm evaluation in robotic simulator

The parameters and algorithm settings (see Tab. 4.6) for the experiments in Ignition Gazebo were set based on the previous results. At first the PK phase ran and estimated 15 and 25 SPs from the prior (reference) model. Both variants were then tested with different total number (25,30,35,40) of SPs in two different model versions: 1) the same model as in Sec. 4.1.2 (see Sec. 4.3.2), and 2) the small disturbance version of the model from Sec. 4.2 (see Sec. 4.3.3).

Parameters		Value
# of SPs		high (15 and 25)
Model division		None
First SP		fixed (Starting position)
# of dimensions		3
Angular res. (hor./ver.)	ter. scanner	0.05° / 0.05°
	LiDAR	0.2° / 0.7°
	label	(Ang comb 0)
Sensor z coordinate		1 m

Table 4.6: Algorithm parameters and settings for simulations in Ignition Gazebo

4.3.1 Initial SPs preparation in PK phase

At the beginning of the PK phase, the 3D prior model was projected into a 2D grid with square cells with a side length of 0.1 m. Then the obstacles were inflated to eliminate possibly infeasible SPs. The inflated 2D grid was subsampled with a given step, and random offsets were added to the new sampled cells. The free samples reachable from the robot start position (a cell with indexes [20,20]) create the set of SPCs. Generated SPCs and the difference between original and inflated grid can be seen in Fig. 4.15. As can be noticed, the grid inflation causes that the SPCs have to be almost always outside the individual robotic cells. Therefore, the scans from the SPs from PK phase provide an overview of the environment without the details hidden due to occlusion. The details have to be gathered during the NBS phase.

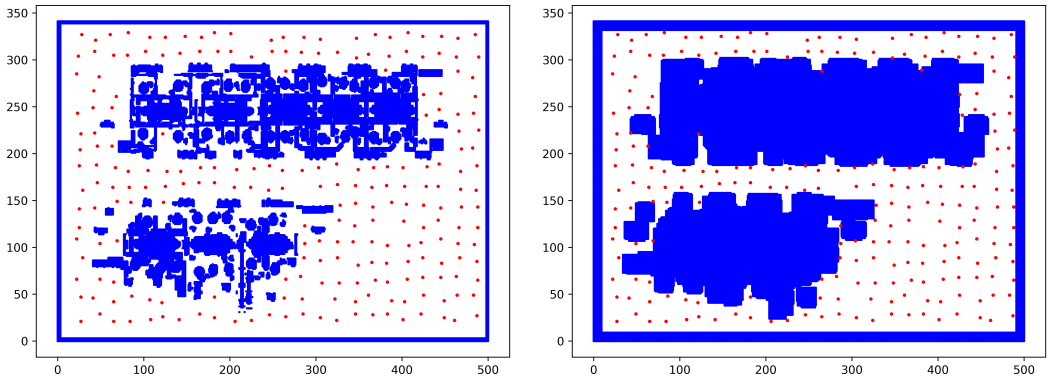


Figure 4.15: 2D projection of the model for the simulation experiments with sampled SPCs (left) and its inflated version (right).

The visibility for each SPC is computed in 3D because the 3D prior model is available. As the terrestrial scanner is used in the experiments, the visibility is computed by casting rays from the model voxels to the SPC. All model voxels in the neighborhood limited by a scanner maximum range are selected, and the direction vectors between these voxels and the SPC are computed.

The first SP is the starting position. The other SPCs are sorted by how much the environment is visible from their positions. Best 14 (24) SPCs were selected as the rest of the SPs. Once the best SPs were chosen, the path between them and their order was determined. As shown in Fig. 4.16, the path for the case with 15 SPs seems to be intuitively worse than the one for the case with 25 SPs, and the order of the SPs is different for the same subset of SPs in both cases.

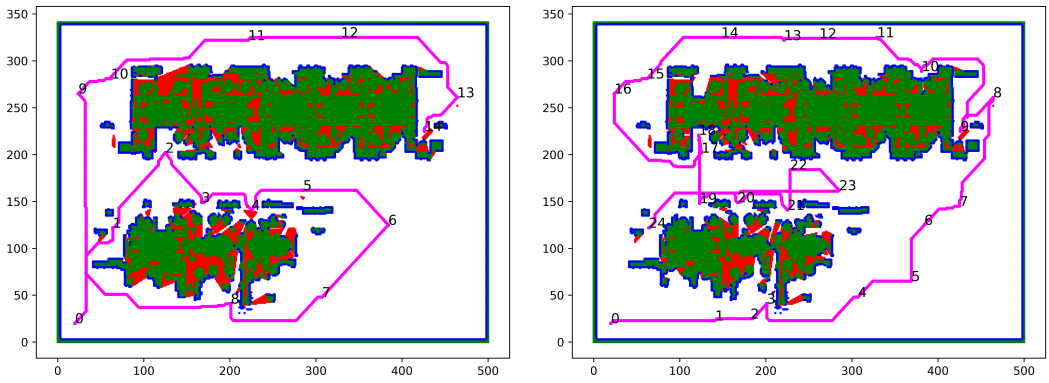


Figure 4.16: Offline phase output for the simulation experiments with the selected SPs and the path traversing them — 15 SPs (left) and 25 SPs (right).

4.3.2 Experiments in the model with 9 shifted objects

At first, the experiments in the model with 9 shifted objects were done to compare the proposed solution with the classical frontier-based exploration (Naex experiments). The gathered point clouds are evaluated after 25, 30, 35, and 40 SPs for two different numbers of SPs prepared in the PK phase — 15 SPs in Exp 1 and 25 SPs in Exp 2. The scan coverage of the environment is shown in Fig. 4.17. As shown in the figure, the environment is fairly covered. Most of the regions are blue (see histograms in Fig. 4.17), which means the majority of the distances are around or less than 1 cm, imply that the environment is now better scanned than during classical frontier-based exploration.

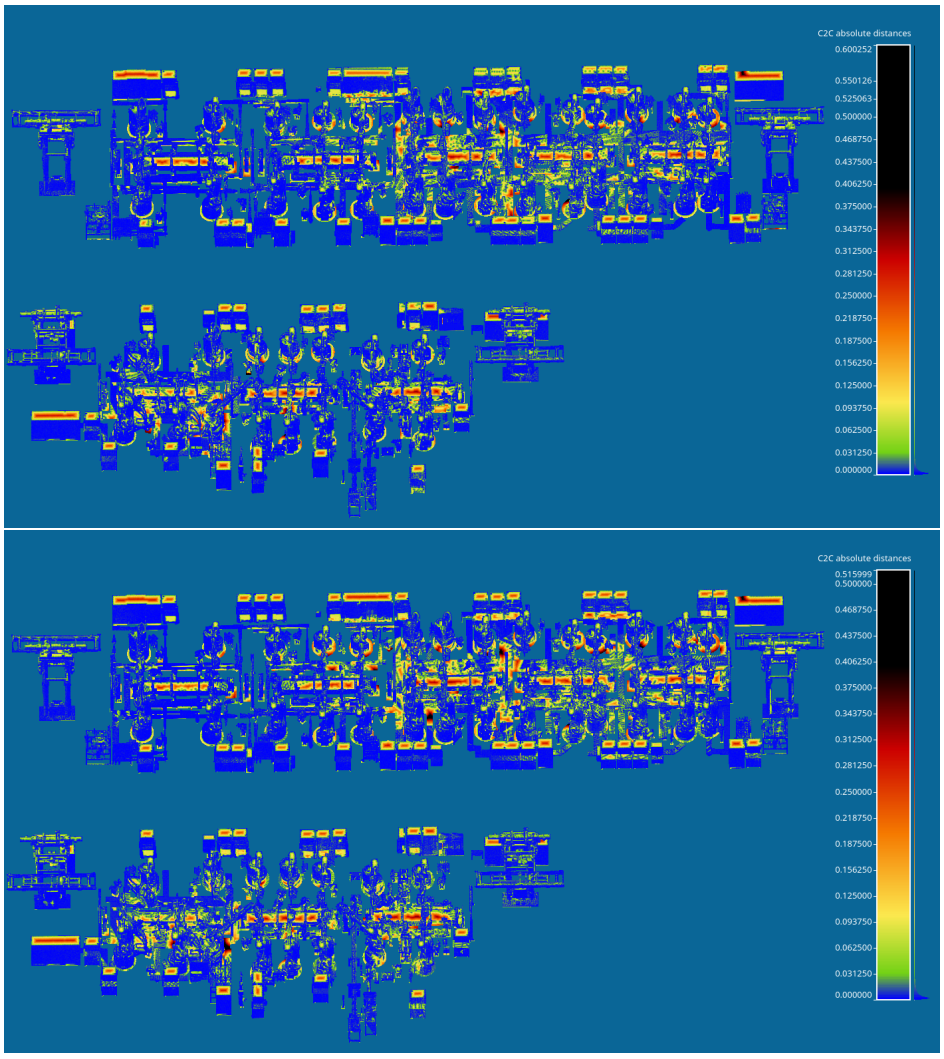


Figure 4.17: Point cloud density as heatmap of distances between reference and measured points in the first environment for Exp 1 (top) and Exp 2 (bottom) experiments. A histogram of distances is on the right side next to the color scale.

Results after object alignment are shown in Fig. 4.18. As can be noticed, the medians (solid lines) remain almost the same for different SPs and their value is about 0.002 m. It means that at least half of the objects are aligned with errors inside an interval ± 0.002 m even for 25

SPs. Furthermore, the figure shows the 75th and 90th percentile errors decrease significantly with increasing number of the SPs, which means that a subset of objects was insufficiently scanned resulting in bad alignment after 25 SPs (errors over 1 cm). The alignment errors for individual objects are shown in Fig. 4.19.

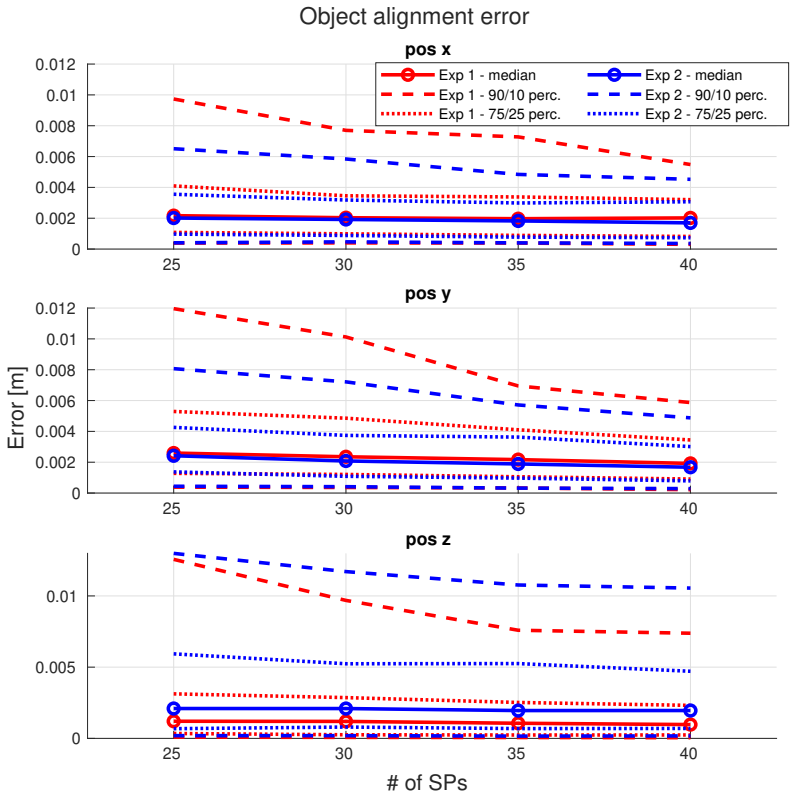


Figure 4.18: Objects positions deviation percentiles in the first environment for Exp 1 and Exp 2 experiments.

The classification was done only in the x-y plane without rotation like in Naex experiments. A decision whether the object was shifted was made regarding the threshold, which was set to the same value for both coordinates. The classification accuracy was computed for four different values of the threshold — 0.005 m, 0.01 m, 0.02 m, 0.05 m. As shown in Fig. 4.20, the accuracy increases with the number of SPs. For thresholds 0.02 m and 0.05 m, it reaches the value of 1 after a certain number of SPs. Furthermore, the accuracy overcomes the accuracy of frontier-based exploration somewhere between 30 and 35 SPs. Moreover, the accuracy for the threshold of 0.005 m is significantly higher, even for 25 SPs. Another important finding is that there are no shifted objects marked as not shifted for all threshold values (i.e., sensitivity is always equal to 1) as can be seen from Tab. 4.7.

The localization error (see Fig. 4.21) is lower than during the Naex experiments, mainly in the problematic z-axis. On the other hand, there are many noisy estimations of the robot's position in the x-axis, which did not appear in the previous experiments. They may be caused by going along the walls and closets.

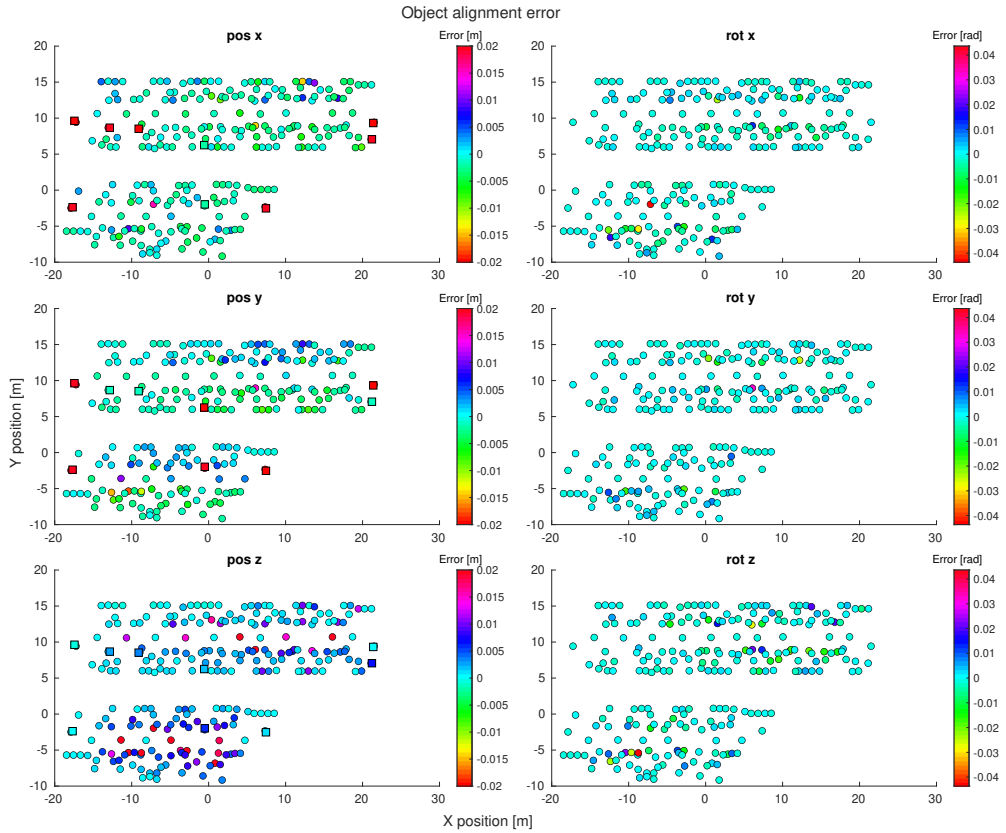


Figure 4.19: Objects positions and computed deviations after 40 SPs in the first environment. Squares represent shifted objects and circles the rest.

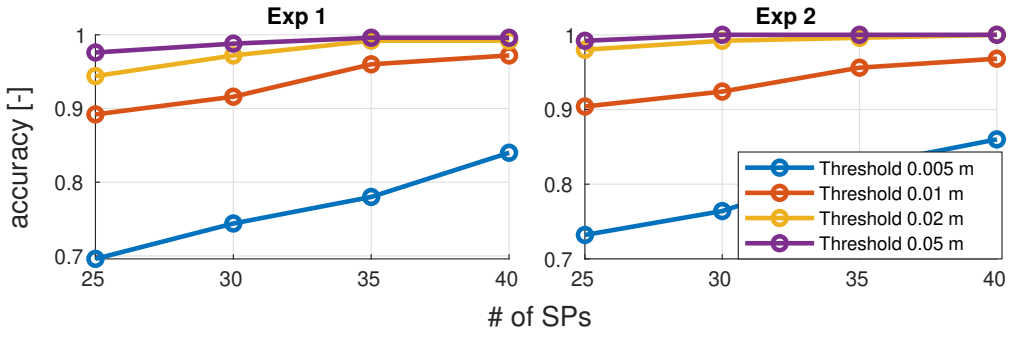


Figure 4.20: Accuracy of shifted objects detection for different threshold values for both experiments in the first environment.

Experiment	threshold							
	0.005 m		0.01 m		0.02 m		0.05 m	
Exp 1 after 25 SPs	165	76	214	27	227	14	235	6
	0	9	0	9	0	9	0	9
Exp 1 after 30 SPs	177	64	220	21	234	7	238	3
	0	9	0	9	0	9	0	9
Exp 1 after 35 SPs	186	55	231	10	239	2	240	1
	0	9	0	9	0	9	0	9
Exp 1 after 40 SPs	201	40	234	7	239	2	240	1
	0	9	0	9	0	9	0	9
Exp 2 after 25 SPs	174	67	217	24	236	5	239	5
	0	9	0	9	0	9	0	9
Exp 2 after 30 SPs	182	59	222	19	239	2	241	0
	0	9	0	9	0	9	0	9
Exp 2 after 35 SPs	197	44	230	11	240	1	241	0
	0	9	0	9	0	9	0	9
Exp 2 after 40 SPs	206	35	233	8	241	0	241	0
	0	9	0	9	0	9	0	9

Table 4.7: Confusion matrices for both experiments in the first environment after given number of SPs. Matrices are written as $\begin{matrix} TN & FP \\ FN & TP \end{matrix}$, where positivity means that objects are shifted.

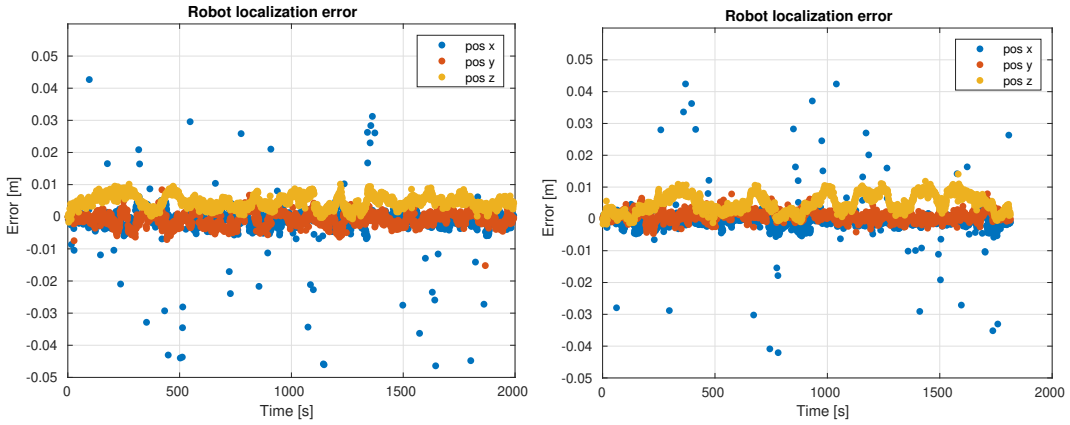


Figure 4.21: Robot estimated position error in the first environment for Exp 1 (left) and Exp 2 (right) experiments.

4.3.3 Experiments in the model with small disturbance

The other experiments were done in the small disturbance model where around 80 objects were shifted using the normal distribution noise (see Tab. 4.3). The gathered point clouds are evaluated after 25, 30, 35, and 40 SPs for two different numbers of SPs prepared in the PK phase — 15 SPs in Exp 3 and 25 SPs in Exp 4. The scan coverage of the environment is shown in Fig. 4.22. The coverage of the environment is less dense than in the previous experiments. Furthermore, there is a remarkable difference between them. The heatmap for Exp 4 is mostly blue, meaning most of the distances are around 1 cm compared to the heatmap for Exp 3, where especially the upper part is mostly green (distances are several centimeters).

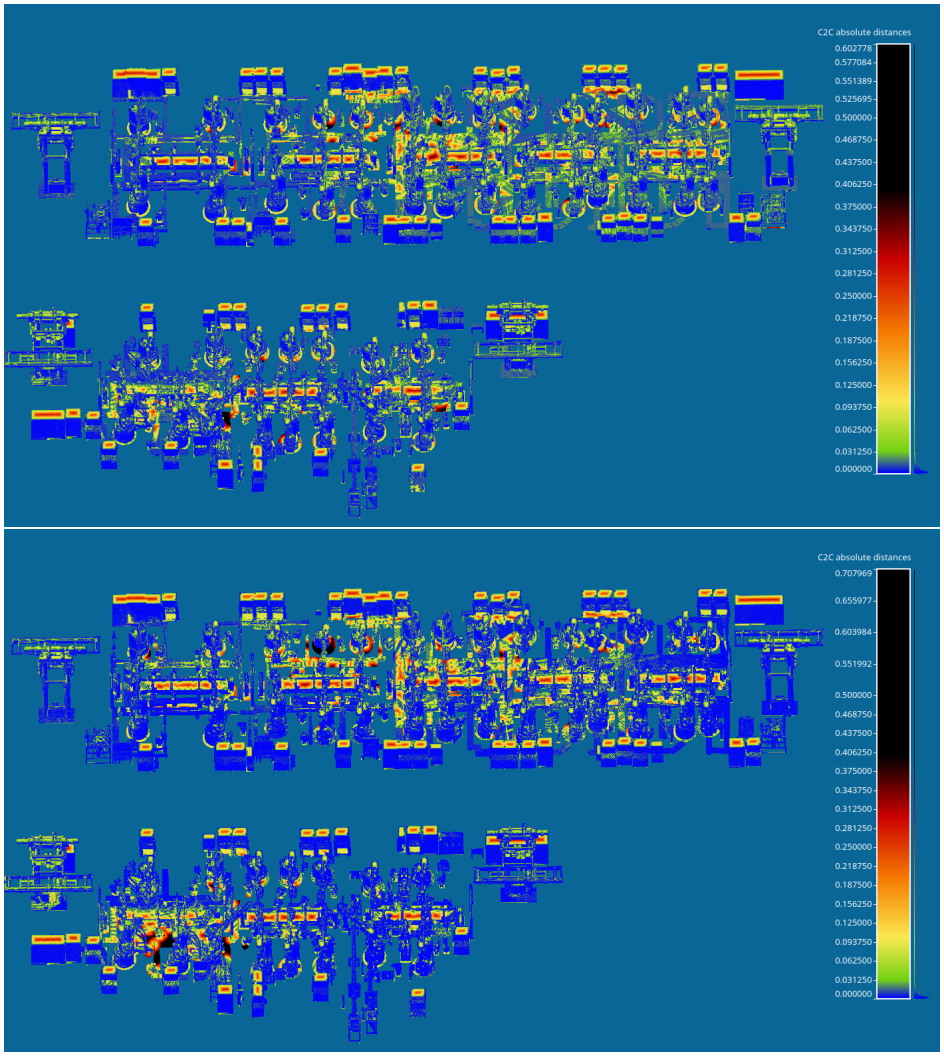


Figure 4.22: Point cloud density as heatmap of distances between reference and measured points in the second environment for Exp 3 (top) and Exp 4 (bottom) experiments. A histogram of distances is on the right side next to the color scale.

The differences between the results of object alignment, shown in Fig. 4.23, reflect the big differences in the coverage of the environment. Medians are lower than 0.006 m for both

experiments. On the other hand, the 90th percentile for Exp 3 is around 0.07 m, i.e., seven times higher than for Exp 4.

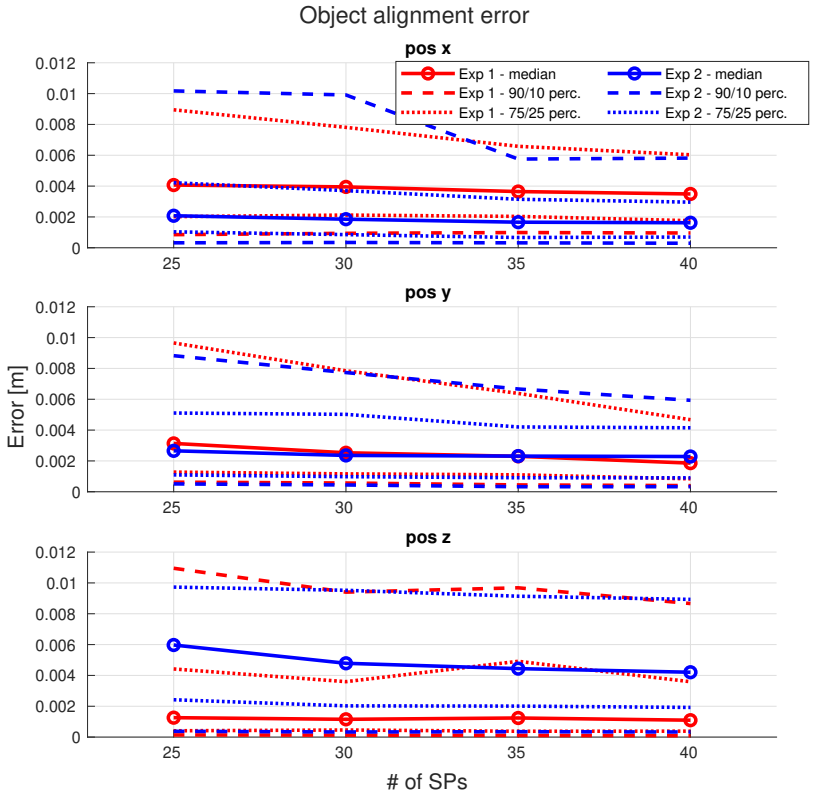


Figure 4.23: Objects positions deviation percentiles in the second environment for Exp 3 and Exp 4 experiments. Value of 90th percentile for Exp 3 is around 0.07 m, i.e., seven times higher than for Exp 4.

In this case, the classification was done for three different variants of ground-truth classes. The variants differ in the border value needed to decide whether the object is really shifted or not. Since the offsets are sampled from a normal distribution, a few of them are too small to be detected. Hence, these should not be taken into account for the classification (i.e., marked as not shifted) as they would affect the results. The border values are 0.01 m, 0.02 m, and 0.05 m. A decision whether the object was shifted was made regarding the threshold, which was set to the same value for both coordinates. The classification accuracy was computed for four different values of the threshold — 0.005 m, 0.01 m, 0.02 m, 0.05 m.

As shown in Fig. 4.24, even for 40 SPs, the accuracy is never equal to 1, the highest values are around 0.95. Furthermore, it can be noticed, the border value affects the accuracy for different threshold values, i.e., the best results are for the same border and threshold value. The classification accuracy is better in Exp 3 than in Exp 4, although the coverage is worse in that case.

The localization error (see Fig. 4.25) is almost the same as in the previous experiments. Again, there are many noisy estimations of the robot's position in the x-axis.

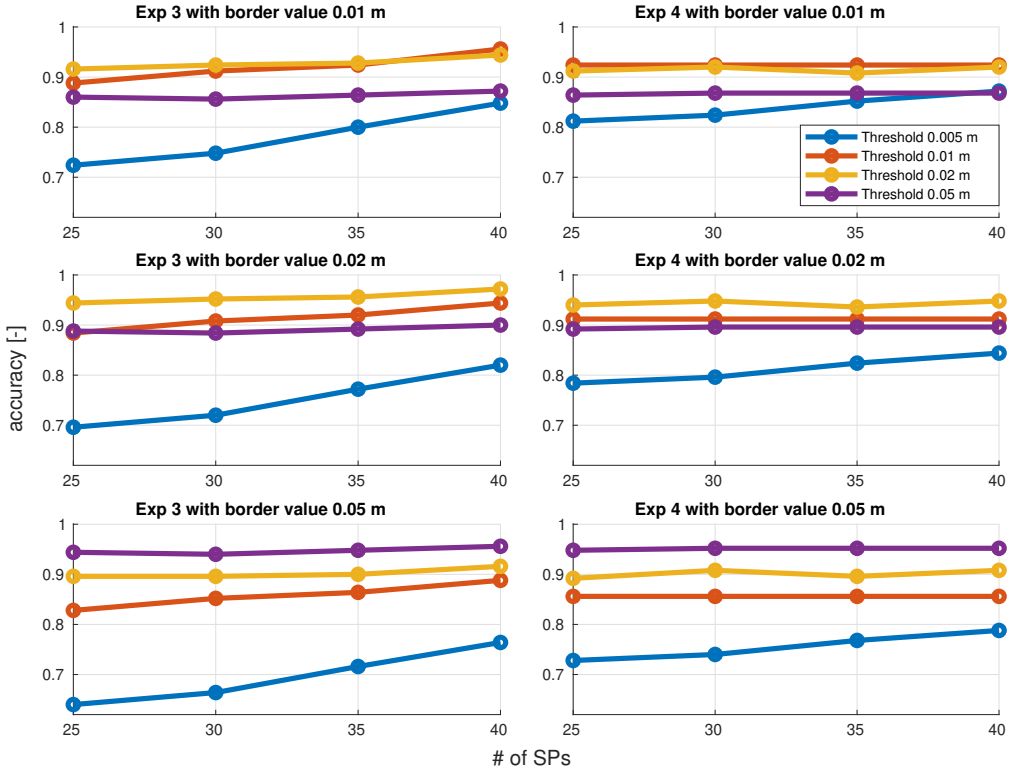


Figure 4.24: Accuracy of shifted objects detection for different threshold values and different border values (rows) for both experiments in the second environment.

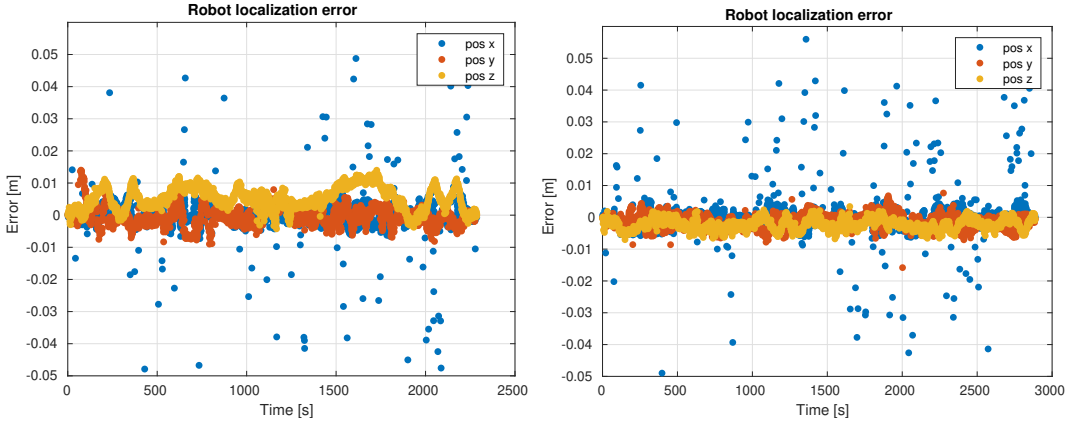


Figure 4.25: Robot estimated position error in the second environment for Exp 3 (left) and Exp 4 (right) experiments.

4.4 Algorithm evaluation in real factory hall environment

Experiments with the real robot (see Sec. 2.2) took place in Testbed. Testbed is an experimental workplace with several robotic cells with different robots. Dimensions of the given model (see Fig. 4.26) are around 55x20x4 m. However, part of the environment is hidden behind the doors, thus inaccessible. As opposed to the factory model, there are no fences between cells, which means the scanner can gather points from the distant robotic cells as well.

Figure 4.26: 3D prior model of Testbed.

Two experiments were prepared. Each of them consisted of 15 SP and lasted about 50 minutes. The first experiment started with the PK phase, whereas the second one tested only the NBS phase of the proposed ESS algorithm. Photos from the experiments are shown in Fig. 4.27. The gathered point clouds were either aligned automatically during the experiment (data from LiDAR for navigation) or manually after the experiment (data from terrestrial scanner as the main output of the experiment). Since there was a problem with the robot localization, the floor in the aligned point clouds gathered by LiDAR was not horizontal, but there was a slope caused by the error in robot estimated position.

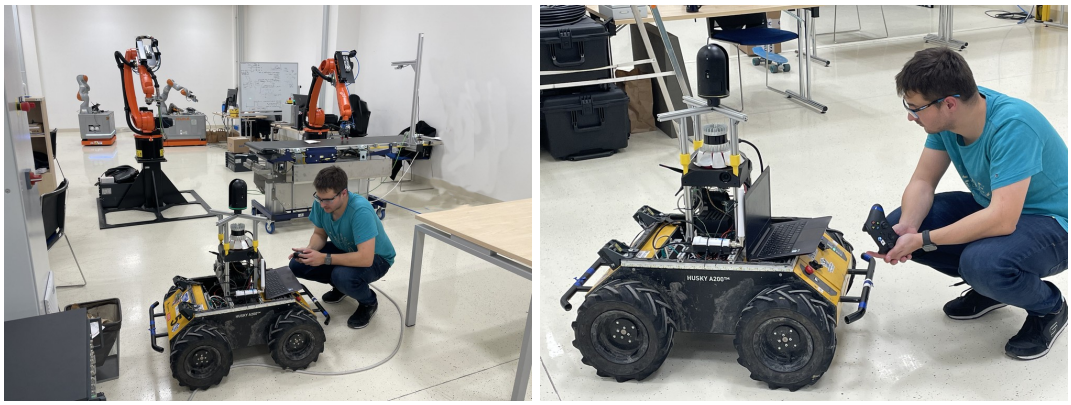


Figure 4.27: Photos from experiment in Testbed taken while the robot stood still and the terrestrial scanner was gathering data. The operator sat next to the robot and held the teleoperation controller.

Experiment with PK phase

At the beginning of the first experiment, the prior model was preprocessed, and the SPCs were sampled (see Fig. 4.28). The big blue rectangle in the right upper corner, meaning the space is occupied, was added manually because no factory equipment was in this part of the area neither in model (empty rooms) nor in reality (leisure area with sofa). As can be noticed, there is no SPC in the right lower corner room. The reason is that there is no path between the main room and this room based on the model (in reality, they are connected through a door). Since the experiment was designed to contain 15 SPs, we decided to prepare 10 SPs in PK phase to cover most of the environment, and the rest of the SP would be used to fill the holes in the model during the NBS phase. The selected SPs and their order can be seen in Fig. 4.28 (right).

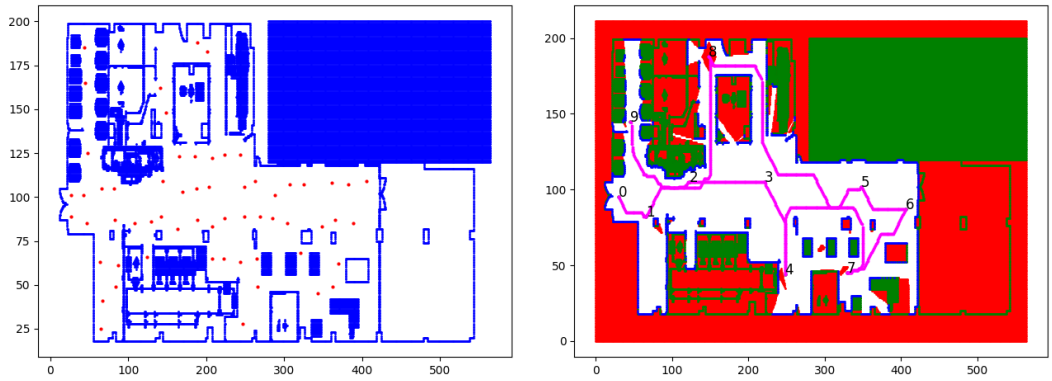


Figure 4.28: 2D projection of Testbed with sampled SPCs (left) and PK phase output, the selected SP with the path traversing them (right).

The final combined point clouds are depicted in Fig. 4.29. As can be seen, the data from the terrestrial scanner were less noisy. On the other hand, the data from the LiDAR were more complete.

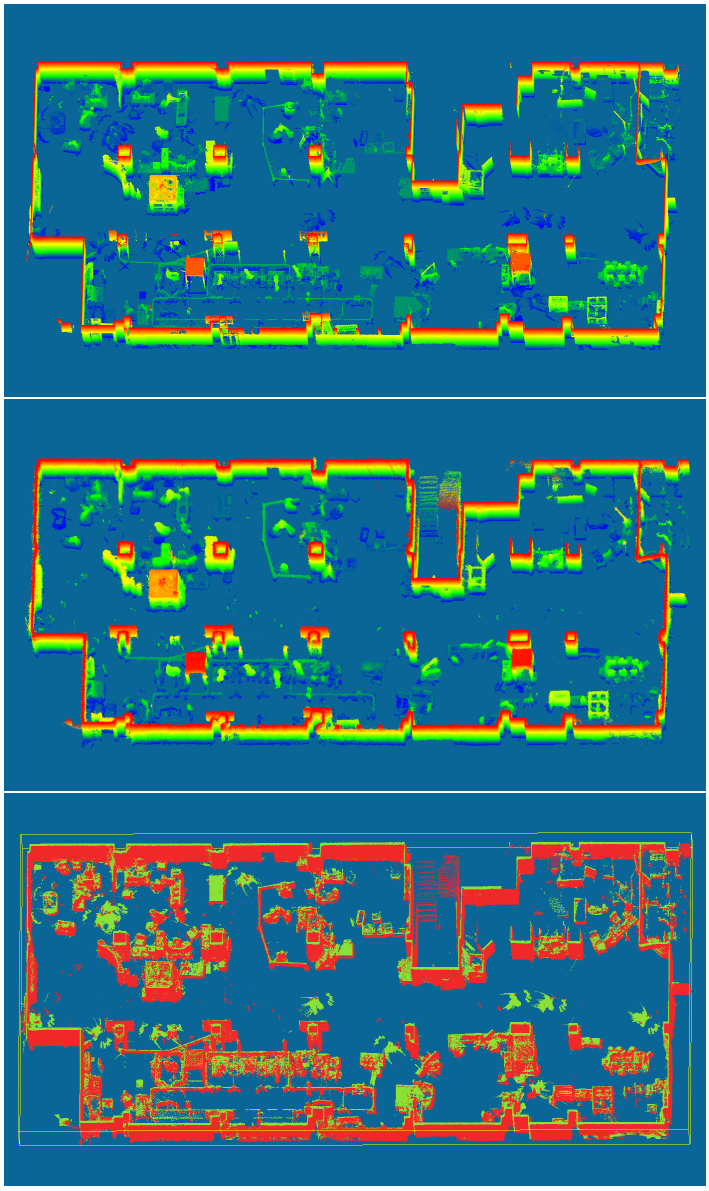


Figure 4.29: Scanned Testbed in the experiment with PK phase by terrestrial scanner (top) and LiDAR (middle) and their overlap (bottom) — terrestrial scanner (green) and LiDAR (red).

Experiment without PK phase

The second experiment consisted of NBS phase only. The reason is that the model did not correspond to the current state (see Sec. 5.2), so the impact of the skipped PK phase was not significant. The final combined point clouds can be seen in Fig. 4.30. Similar to the previous experiments, the data from the terrestrial scanner were less noisy. On the other hand, the data from the LiDAR were more complete.

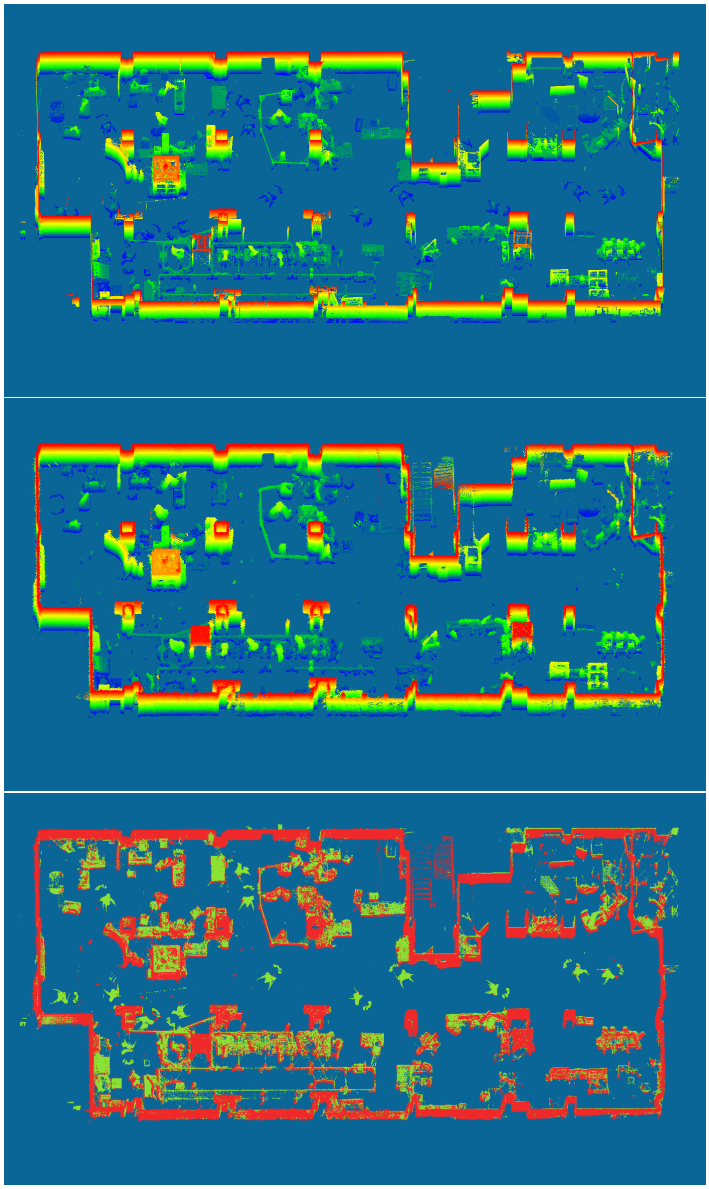


Figure 4.30: Scanned Testbed in the experiment without PK phase by terrestrial scanner (top) and LiDAR (middle) and their overlap (bottom) — terrestrial scanner (green) and LiDAR (red).

Experiments comparison

Since both experiments lasted a similar time and the number of scans was the same, the results should be similar. As can be seen in the detailed view of the gathered data (see Fig. 4.31), the right bottom corner of the model is better covered in the first experiment. Nevertheless, when Figures 4.29 (top) and 4.30 (top) are compared, the differences in the rest of the model are not significant.

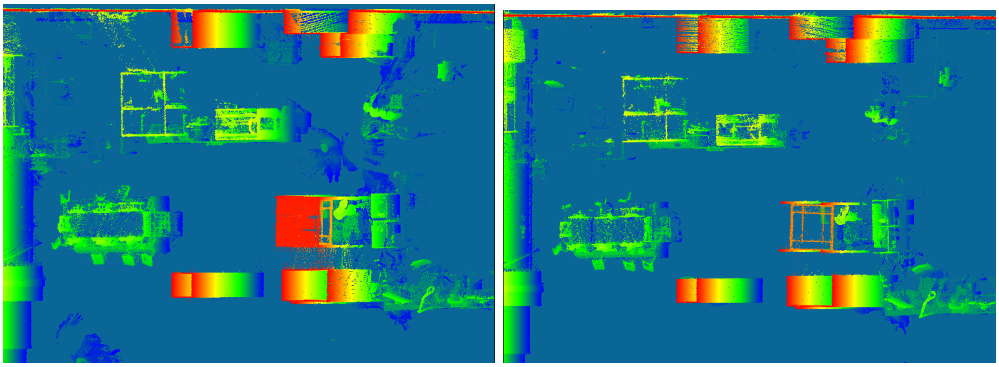


Figure 4.31: Detailed view on the gathered data in the right bottom corner of the model. First (left) and second (right) experiment in Testbed.

Furthermore, the traversed paths are compared in Fig. 4.32. The robot stayed on the long passage in the center during the second experiment (see Fig. 4.32 bottom). Once the robot was going back from the far side, the SPs were planned closer to objects and walls. The SPs planned in the PK phase of the first experiment (see Fig. 4.32 top) were placed near the objects. The robot could not reach the desired positions because the SPs were planned on the misleading model. Nevertheless, as seen earlier, even these inaccurate SPs improve the model coverage.

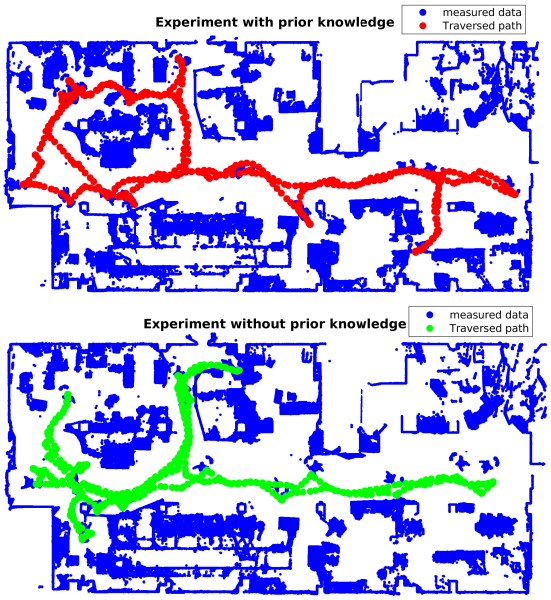


Figure 4.32: Traversed path during the experiments in Testbed with prior (top) and without (bottom) knowledge.

Chapter 5

Discussion

In this chapter, we discuss the problems we faced during the implementation (Sec. 5.1) and the real-world testing (Sec. 5.2). Then we summarize the obtained knowledge in a study of technical feasibility in Sec. 5.3.

5.1 Implementation issues

During implementation of Exploration by Static Scans (ESS) algorithm, several issues appeared. Therefore, several changes were made to resolve these issues or at least to minimize their effect.

The first issue was related to the problem of the gathered high-resolution point clouds (un)availability during the experiment. Since the output static scans could not be used for the next Scanning position (SP) decision, these gathered scans had to be approximated. In the first implementation, the output scans (and the world model) had been estimated by the scans collected by the LiDAR in the SP. An approximation method was introduced (see Sec. 3.4.3) because of the differences between the scans from the terrestrial scanner and the LiDAR.

The second issue was related to the gain computation function. Originally, cast rays had been sampled with high angular resolution in both directions. However, the ray-casting computation with high resolution had been computationally demanding. Moreover, the high resolution had influenced the coverage of far voxels only because nearby voxels had been traced by multiple rays, which had been unnecessary and ineffective. Therefore, the angular resolution of the rays was changed (see Sec. 3.4.5).

The last issue was that there was currently no possibility to automate the static scan process. This issue appeared only in the real-world scenario. There was no communication between the program and the terrestrial scanner, which meant that the scanning procedure had to be started manually and the program could not detect that the scan was finished. In the first version, the program had waited for a specific amount of time, and when the time had been exceeded, the robot had started moving again. However, looking for a suitable time limit had not been trivial, because the scanning procedure had been started manually by the operator. Therefore, instead of the time limit, a specific button had to be pressed on the teleoperation controller when the scan was finished (more details in Sec. 3.5.2).

5.2 Real-world experiments issues

Several problems occurred during the real-world experiments in Testbed. The first one was polished floor, which caused difficulties for the navigation program and the ESS algorithm.

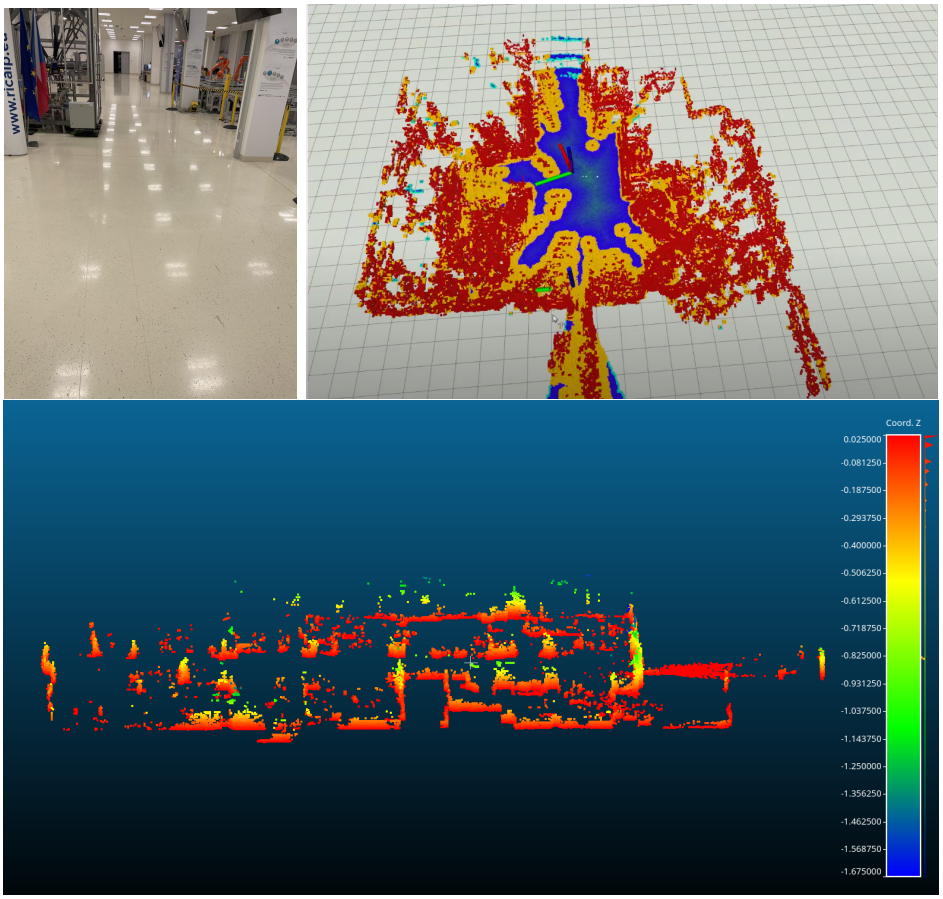


Figure 5.1: The polished floor in Testbed with visible light reflection (top left), the created artifacts (orange circles with red dots) in the navigation map (top right), and gathered points under the ground level (bottom).

A problematic (light) ray reflection can be seen in Fig. 5.1 (top left). Like light rays, laser rays from the LiDAR were also reflected. As a result, the reflected rays caused fragments in the navigation map — the orange circles with red dots shown in Fig. 5.1 (top right). Due to the reflection, the navigation algorithm assumed that these artifacts were holes in the floor and the robot could not pass through them. Therefore, the robot movement was sometimes chaotic as the fragments were constantly changing. The reason is that many measured points seemed to be under the floor, mainly for the measurements close to the walls and closets as the reflected rays from the floor hit these obstacles and then went back to the scanner, resulting in a higher measured distance compared to reality. It can be observed in Fig. 5.1 (bottom), where the measured points under the floor are shown. Furthermore, the robot localization estimation was disturbed (see deviation of z-axis coordinate of the robot estimated position during experiments in Fig. 5.2).

The second problem was the inaccurate prior model. As shown in Fig. 5.4, the measured data and the given model were not similar. Even if the manipulators were in both models (the given one and the measured one), they often had different joint configurations, making the object alignment more difficult. Since there was no ground truth, the evaluation of the experiments could not be done properly. On the other hand, it could be used to test whether

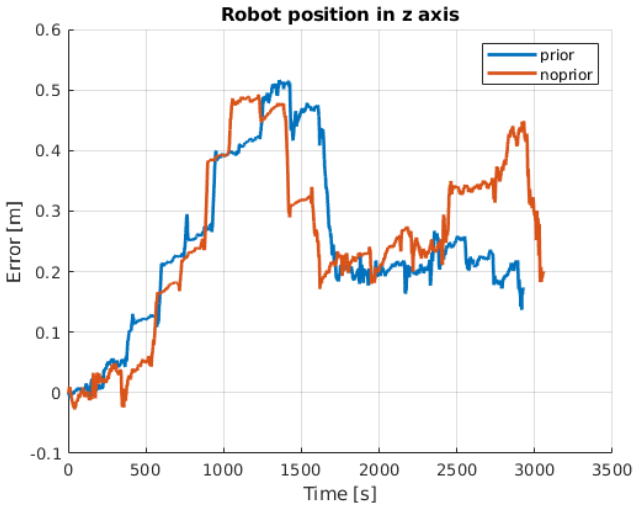


Figure 5.2: The change of the robot position in z-axis from the starting position during experiments.

the proposed algorithm was able to deal with unreachable scanning poses.

Another observed problem was that the implemented alignment method does not work in this case (see Fig. 5.3), probably because of the initialization. Unfortunately, our alignment method was sensitive to the quality of initial estimation. The alignment likely failed once the objects were not close to the correct position or their shape was not the same.

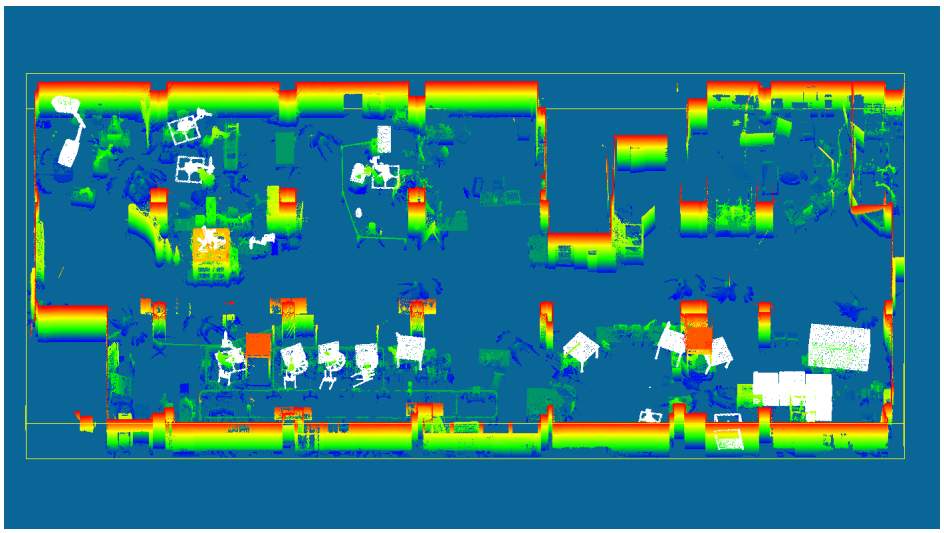


Figure 5.3: An unsuccessful attempt to align objects with measured data.

5.3 Study of technical feasibility

In this section, we summarize the prepared method for automated 3D scanning in factory halls, highlight the achieved results, discuss the drawbacks and problems, and propose possible

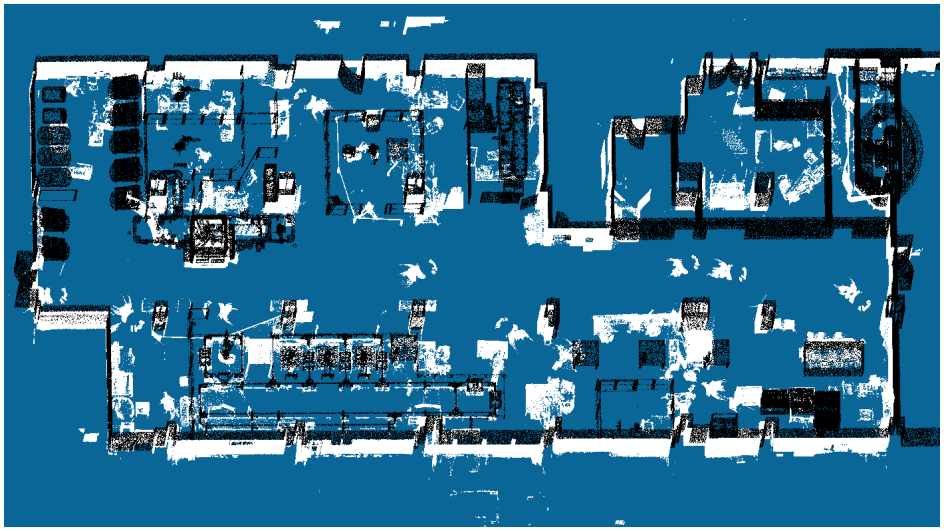


Figure 5.4: Inaccurate 3D prior model (black) vs the actually measured model (white).

enhancements.

5.3.1 Exploration algorithm

The original frontier-based exploration algorithm had to be replaced by the new exploration algorithm because the high-resolution terrestrial scanner could collect data only while it stood still. The proposed exploration algorithm called ESS maximizes the coverage of the unknown environment and minimizes the required time of the process, i.e., the number of SPs. After the experiments, we can conclude that the division into two phases is useful—the environment is explored more effectively (see Fig. 4.14). As shown in Figures 4.17 and 4.22, the whole environment was covered. Gathered points were closer than 1 cm to points of the corresponding objects. The coverage was more dense than in the experiments without the terrestrial scanner (see Figures 4.2 and 4.6) as expected. The measured data from the terrestrial scanner were almost without noise; hence the shifted object classification accuracy achieved values close to 1, mainly for the experiments with the shift of at least several centimeters. The classification of shifted objects was not successful only when the area was not covered properly as can be seen in Fig. 4.22. In the case of the real-world experiment, we had no ground-truth data. Nonetheless, it visually seemed the environment was sufficiently covered.

The static model approximation during the factory mapping process worked fine, mainly because it exploited that the first SPs were precomputed so that it was not necessary to approximate the model from the beginning of the process. Nevertheless, it would be better if the data from the terrestrial scanner were available during the process and incorporated into the model as soon as possible.

The main drawback of the proposed algorithm is that it assumes the SPs only in a 2D grid. However, the factory floor can be in different height levels, e.g., a platform accessible by stairs. Once the selected mobile robot is able to reach these areas, it will be unavoidable to change the selection of Scanning position candidates (SPCs) from 2D to 3D. Furthermore, representation

of the scanned model should be changed or customized to improve and accelerate removal of the dynamic objects and improve empty voxel estimation. Moreover, the navigation algorithm should be closely connected with the exploration algorithm to detect unreachable SPCs effectively and improve the robot's path planning.

The robot's return to the dock station can arise several problems. It is necessary to resolve the (automated) model division to suspend the factory scanning process between two parts and not when the current part is not finished. The model division should be done with respect to the robot's remaining power and the distance from the dock station. Another problem would be precise detection of the docking/charging station.

Finally, we suggest several other possible features and future enhancements. The first one is automated detection of robotic cells, which would be helpful to divide the environment automatically and would enable the processing of cells independently. It would be necessary to detect the doors and exploit the found fences to limit the scanning area. Another feature would be reporting of unreachable areas of the environment, which should speed up the process by skipping additional attempts to reach these areas. If there is no prior model or the prior model is inaccurate, we suggest letting the robot traverse the environment rapidly without using the terrestrial scanner and gathering an approximate model of the environment for the Prior Knowledge (PK) phase.

Furthermore, the procedure could be sped up and improved by object recognition and classification online during the scanning process. It would help to identify the object interiors that cannot be seen and distinguish them from the uncovered parts of the objects. The static model could be improved by incorporating the postprocessed measured data by terrestrial scanner during the robot's charging in the dock station.

5.3.2 Autonomous robot behavior

The robot's autonomous behavior is ensured by programs from the framework RDS [27, 28, 29, 31], using norlab-icp-mapper [24] for robot localization estimation. The robot traversed the environments successfully both in simulation and real-world scenarios. However, sometimes it struggled in narrow passages because the navigation algorithm wrongly assumed the robot could not move through due to map resolution and LiDAR noise. Since the navigation program is parametrized to be adaptive to different environments, it is necessary to find suitable values for various settings to improve the navigation performance and robustness. However, the optimal parametrization must be found in the actual environment with all aspects considered.

The localization error evaluated in the simulated experiments can be seen in Figures 4.5, 4.9, 4.21, and 4.25. The localization error in the z-axis was problematic during the classical frontier-based exploration (Naex) experiments because the errors were about 0.03 m which resulted in a sloping floor. On the other hand, there were noisy estimations of the robot's position in the x-axis during the proposed algorithm exploration, probably caused by going along the walls and closets. The more significant problems with localization were in the real-world scenario, where the errors were over 0.4 m in the z-axis. This problem was discussed in Sec. 5.2. For the actual deployment, we assume that the floor in the real factory will not be polished to a mirror-like appearance. Otherwise, the LiDAR data postprocessing method should be implemented. Moreover, we suggest placing position calibration marks (e.g., QR

codes) in different places in the factory hall, such as walls and robotic cell doors, to stabilize the robot localization.

5.3.3 Hardware platform

The scanning mobile robot platform prototype was tested only in Testbed, not in a real factory environment[1]. Therefore, the experiment outputs may not reflect the issues which would appear in the factory. The Husky robot traversed the environment reliably. It safely dealt with minor floor roughness and obstacles, such as small robot pedestals and Ethernet cables. However, the four-legged robots seem to be more suitable for the factory environment since they have better traversability. Specifically, they can overcome stairs and obstacles such as cable channels distributing electricity. Moreover, they can better move in narrow areas.

The terrestrial scanner gathered high-resolution data covering the environment densely as can be seen in Figures 4.29-4.30. The problem of missing connection between the scanner and the control algorithm remains, due to which the human operator must be close to the scanner and manually start scanning. It leads to occlusion caused by the human operator and to contamination of the gathered point clouds with points corresponding to the human operator. However, it should be possible to purchase an SDK from Leica to control the scanner directly from the robot. Moreover, the terrestrial scanner should be powered from the robot and not externally with batteries. The platform should work with the remaining power and decide when it is necessary to return to a dock station.

It is possible to incorporate a scanner with even higher precision and points density than the Leica BLK360, such as Leica RTC360. Nevertheless, its weight is several times higher; hence the robot's permissible payload must be considered. Moreover, it should be more deeply studied whether the higher points density is needed and valuable.

[1] Experiments in a real factory are planned just after the submission deadline.

Chapter 6

Conclusion

In this thesis, we proposed a method of autonomous 3D scanning in factory halls with a mobile robotic platform. Mainly, we prepared a new exploration algorithm called Exploration by Static Scans (ESS) optimizing the completeness of 3D data needed for the installation check in combination with minimizing the needed time for the process. Several tasks had to be done to test the proposed algorithm.

At first, we prepared several metrics to evaluate the ESS algorithm. The first metric is used to select the most suitable variants of the ESS algorithm for maximizing the environment coverage. The other metrics are employed to verify the ESS algorithm performance in optimizing the completeness of 3D data needed for the installation check.

Furthermore, we assembled a hardware prototype of a scanning mobile robotic platform to test the ESS algorithm in a real-world scenario. The platform is based on the robot setup for DARPA SubTerranean Challenge, and it consists of a four-wheeled mobile robot Husky, an ultra-wide view LiDAR sensor Ouster os-0, and other sensors for robot localization and navigation. We extended the setup with a terrestrial scanner Leica BLK360 provides high-resolution scans. Custom construction for attaching the Leica scanner to the robot was designed and the Leica scanner was mounted above the Ouster LiDAR. Then we verified its functionality together with the behavior of the ESS algorithm in a real factory hall environment.

The proposed ESS algorithm consists of two phases. Since a very high scan precision is required, the scans are gathered only when the robot stands still. Scanning positions (SPs) are positions where these high-resolution scans are gathered. The goal of the algorithm is to find optimal SPs to maximize the coverage of the unknown environment. The algorithm may exploit the prior knowledge about the environment (if available) to speed up the exploration procedure by computing several SPs ahead, i.e., before the exploration starts.

As we assume that the data from the terrestrial scanner are not available during the factory mapping process, it is necessary to estimate the proportion of the environment the terrestrial scanner has already scanned. Therefore, we proposed an approximation method for estimating the gathered high-resolution point clouds from terrestrial scanners. The approximation exploits the information about the environment gathered by the LiDAR.

We performed several experiments to test the performance of our proposed algorithm. At first, we ran classical frontier-based exploration experiments to obtain baseline results and test the prepared metrics. The robot was gathering data simultaneously during these experiments with the LiDAR. The majority of the distances between the gathered data and the reference point cloud were a few centimeters. Nevertheless, the alignment error was within ± 0.02 m for almost all objects, and most of them were within ± 0.01 m for the x-axis and y-axis. The small alignment errors led to satisfactory classification results — the recall always equals 1,

and the accuracy equals 1 for the object position deviation threshold 0.02 m and 0.05 m.

We conducted experiments to choose the best variant of our ESS algorithm. Based on the similarity score computation, a prior model should be used because the coverage of the environment increases slowly without the prior model. Moreover, a 3D prior model should be preferred instead of a 2D prior model if both are available. Another outcome is that it is favorable not to divide the environment into parts but to process it at once.

The evaluation of the proposed algorithm in the robotic simulator was performed in two different versions of the model, each of them with a different number of shifted objects. However, the computation of SPs in Prior Knowledge (PK) phase was done in the same prior model different from them. We prepared two sets of SPs prepared in PK phase with 15 and 25 SPs respectively. Then during the experiments, scans were gathered in 40 SPs. In the first model version, the environment was fairly covered. The majority of the distances between the gathered data and the reference point cloud were around 1 cm. The object alignment errors in the x-axis and y-axis were promising as well. Their mean was around 0.002 m even after 25 SPs and the 90th percentile was less than 0.006 m after 40 SPs. The classification accuracy equals 1 for threshold 0.02 m from 35 SPs, and the recall is 1 for all thresholds from 25 SPs.

In the second model version, there is a difference between the environment coverage for the cases with 15 and 25 SPs prepared in PK phase, mainly in the top part of the model. The difference propagated into the object alignment errors in the x-axis and y-axis. The means were around 0.004 m and 0.002 m for 15 and 25 SPs prepared in PK phase respectively. The value of 90th percentile was more than seven times higher for the case with 15 SPs than with 25 SPs prepared in PK phase. The classification accuracy is over 0.9 for the threshold of 0.02 m and all border values. In contrary to the object alignment results, the classification results have higher accuracy for the case with 15 SPs than with 25 SPs prepared in PK phase.

Finally, we tested the algorithm in a real-world scenario in Testbed environment. Unfortunately, the prior model of the environment does not correspond to the current state; hence, it is not possible to evaluate the measured data using our metrics. We ran two experiments — with and without the prior model. For each experiment, scans were collected in 15 SPs. During the experiments, we discovered that the floor in Testbed was polished to a mirror-like appearance. The reflected rays caused fragments in the navigation map and added noise to the robot localization estimation, leading to errors around 0.4 m in the robot position in the z-axis. Despite this problem, the robot successfully gathered all scans and returned to the starting position.

As a final task, we summarized the obtained knowledge of using an autonomous robot to automate 3D scanning in factory halls in a study of technical feasibility. In the study, we evaluated the combination of the robotic scanning platform prototype and the proposed algorithm. We identified the drawbacks of the current solution and proposed possible improvements for actual deployments.

Bibliography

[1] K. Klein and V. Sequeira, "View planning for the 3D modelling of real world scenes," *IEEE International Conference on Intelligent Robots and Systems*, vol. 2, 2000.

[2] A. Adán, B. Quintana, and S. A. Prieto, "Autonomous mobile scanning systems for the digitization of buildings: A review," 2019.

[3] B. Charrow, G. Kahn, S. Patil, S. Liu, K. Goldberg, P. Abbeel, N. Michael, and V. Kumar, "Information-theoretic planning with trajectory optimization for dense 3D mapping," in *Robotics: Science and Systems*, vol. 11, 2015.

[4] R. Kurazume, S. Oshima, S. Nagakura, Y. Jeong, and Y. Iwashita, "Automatic large-scale three dimensional modeling using cooperative multiple robots," *Computer Vision and Image Understanding*, vol. 157, 2017.

[5] C. Potthast and G. S. Sukhatme, "A probabilistic framework for next best view estimation in a cluttered environment," *Journal of Visual Communication and Image Representation*, vol. 25, no. 1, 2014.

[6] 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*, vol. 10, no. 4, 2015.

[7] F. Bissmarck, M. Svensson, and G. Tolt, "Efficient algorithms for Next Best View evaluation," in *IEEE International Conference on Intelligent Robots and Systems*, vol. 2015-December, 2015.

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

[9] S. Song, D. Kim, and S. Jo, "Online coverage and inspection planning for 3D modeling," *Autonomous Robots*, vol. 44, no. 8, 2020.

[10] Z. Meng, H. Qin, Z. Chen, X. Chen, H. Sun, F. Lin, and M. H. Ang, "A two-stage optimized next-view planning framework for 3-D unknown environment exploration, and structural reconstruction," *IEEE Robotics and Automation Letters*, vol. 2, no. 3, 2017.

[11] P. S. Blaer and P. K. Allen, "Data acquisition and view planning for 3-D modeling tasks," in *IEEE International Conference on Intelligent Robots and Systems*, 2007.

[12] P. Kim, J. Chen, and Y. K. Cho, "SLAM-driven robotic mapping and registration of 3D point clouds," *Automation in Construction*, vol. 89, 2018.

[13] A. Adán, B. Quintana, A. S. Vázquez, A. Olivares, E. Parra, and S. Prieto, "Towards the automatic scanning of indoors with robots," *Sensors (Switzerland)*, vol. 15, no. 5, 2015.

[14] N. Palomeras, N. Hurtos, E. Vidal, and M. Carreras, "Autonomous exploration of complex underwater environments using a probabilistic next-best-view planner," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, 2019.

[15] L. M. González-de Santos, L. Díaz-Vilariño, J. Balado, J. Martínez-Sánchez, H. González-Jorge, and A. Sánchez-Rodríguez, "Autonomous point cloud acquisition of unknown indoor scenes," *ISPRS International Journal of Geo-Information*, vol. 7, no. 7, 2018.

[16] Center for Robotics and Autonomous Systems, "Ctu-cras-norlab team for darpa sub-t." `http://robotics.fel.cvut.cz/cras/darpa-subt/`.

[17] DARPA, "DARPA SubTerranean challenge." `https://www.subtchallenge.com/`.

[18] Stanford Artificial Intelligence Laboratory et al., "Robotic operating system." `https://www.ros.org`.

[19] Open Robotics, "Ignition robotics." `https://ignitionrobotics.org`.

[20] Clearpath Robotics, "Mobile robots for research & development." `https://clearpathrobotics.com/`.

[21] Boston Dynamics, "Spot: Boston dynamics." `https://www.bostondynamics.com/spot`.

[22] M. Pecka, "Ctu-cras-norlab husky sensor config 1." `https://github.com/osrf/subt/blob/final_event/submitted_models/ctu_cras_norlab_husky_sensor_config_1/specifications.md`.

[23] Google, "Protocol Buffers | Google Developers." `https://developers.google.com/protocol-buffers/`.

[24] Norlab Ulaval, "Norlab icp mapper." `https://github.com/norlab-ulaval/norlab_icp_mapper`, 2021.

[25] P. Besl and N. D. McKay, "A method for registration of 3-d shapes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, no. 2, pp. 239–256, 1992.

[26] K. S. Arun, T. S. Huang, and S. D. Blostein, "Least-squares fitting of two 3-d point sets," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, no. 5, pp. 698–700, 1987.

[27] J. Bayer, "Robot deployment system." `https://gitlab.fel.cvut.cz/bayerja1/robot_deployment_system`, 2021.

[28] J. Bayer and J. Faigl, "Decentralized topological mapping for multi-robot autonomous exploration under low-bandwidth communication," in *2021 European Conference on Mobile Robots (ECMR)*, pp. 1–7, 2021.

[29] J. Bayer and J. Faigl, "Decentralized task allocation in multi-robot exploration with position sharing only," in *International Symposium on Swarm Behavior and Bio-Inspired Robotics (SWARM)*, 2021.

[30] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische mathematik*, vol. 1, no. 1, pp. 269–271, 1959.

[31] J. Bayer and J. Faigl, "On autonomous spatial exploration with small hexapod walking robot using tracking camera intel realsense t265," in *2019 European Conference on Mobile Robots (ECMR)*, pp. 1–6, 09 2019.

[32] T. Petricek, "Naex." `https://gitlab.fel.cvut.cz/cras/subt/common/naex`, 2021.

[33] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "OctoMap: An efficient probabilistic 3D mapping framework based on octrees," *Autonomous Robots*, 2013.

[34] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "OctoMap: An Efficient Probabilistic 3D Mapping Framework Based on Octrees," *Autonomous Robots*, 2013. Software available at `http://octomap.github.io`.

[35] A. Hornung, "Octovis." `https://github.com/OctoMap/octomap`.

[36] J. Bresenham, "A linear algorithm for incremental digital display of circular arcs," *Communications of the ACM*, vol. 20, no. 2, 1977.

[37] M. L. V. Pitteway, "Algorithm for drawing ellipses or hyperbolae with a digital plotter," *The Computer Journal*, vol. 10, pp. 282–289, 1967.

[38] J. Bresenham, "Algorithm for computer control of a digital plotter," *IBM Systems Journal*, vol. 4, pp. 25–30, 1965.

[39] J. Amanatides and A. Woo, "A fast voxel traversal algorithm for ray tracing," *Proceedings of EuroGraphics*, vol. 87, 08 1987.

[40] Alphathon, "Diagram of an xbox 360 controller with button labels." `https://commons.wikimedia.org/wiki/File:360_controller.svg`, 2010.

[41] D. Girardeau-Montaut, "Cloudcompare." `https://cloudcompare.org`.

[42] Czech Institute of Informatics, Robotics and Cybernetics, "Testbed for industry 4.0." `https://www.ciirc.cvut.cz/teams-labs/testbed/`.

[43] FileFormat, "Jt file format." `https://docs.fileformat.com/3d/jt/`.

[44] Siemens, "Tecnomatix - siemens digital industries software." `https://www.plm.automation.siemens.com/global/en/products/tecnomatix/`.

[45] Khronos Group Inc., "Collada (dae) file format." `https://www.khronos.org/files/collada_spec_1_5.pdf`.

[46] Wavefront, "Wavefront obj file format." `http://fegemo.github.io/cefet-cg/attachments/obj-spec.pdf`.

[47] Blender Online Community, *Blender - a 3D modelling and rendering package*. Blender Foundation, Stichting Blender Foundation, Amsterdam, 2018.

[48] P. Cignoni, M. Callieri, M. Corsini, M. Dellepiane, F. Ganovelli, and G. Ranzuglia, "MeshLab: an Open-Source Mesh Processing Tool," in *Eurographics Italian Chapter Conference* (V. Scarano, R. D. Chiara, and U. Erra, eds.), The Eurographics Association, 2008.

Appendix A

Model preprocessing

This appendix documents the factory model preprocessing. The model was converted into human-readable file formats, then analyzed and simplified. Finally, we prepared programs and a configuration file to generate environments with different objects or different poses of objects.

A.1 Data representation

The model of the environment was represented in several ways and formats in the model preparation pipeline.

Original models are stored in a .jt format created by Siemens PLM Software [43] as a default export format for Siemens Digital industries software [44]. This format can contain facet data, product and manufacturing information, and metadata in binary form. Although .jt format was officially declared ISO-standardized 3D visualization format, it can be opened only in specific programs and data in binary form are not easily editable. For that reason, it was necessary to convert it to human-readable formats supported by robotic simulators.

COLLADA format (.dae) is a file format for interactive 3D applications with XML-based schema [45]. Models can be stored in a tree structure which simplifies the reading of the file. Moreover, individual blocks, e.g., robotic arms, can be easily parsed and extracted from the whole model.

Wavefront object file format (.obj) is a data format for 3D geometry definition [46]. The object file usually contains vertex positions, vertex normals, texture vertices, and the faces that form each polygon as a list of vertices and their normals.

A models in the .obj and .dae file formats is represented as a polygonal mesh - a collection of vertices, edges, and faces that make up the shape of a polyhedral object. Later, the meshes were converted into **voxels** (i.e., voxelized) - a grid of cubic volumes of equal size - for easy manipulation and evaluation in the exploration algorithm. **Octree** models (see Sec. 3.1) were obtained from the voxelized models.

A mesh can be easily converted to a point cloud by sampling its surface. The model represented by a point cloud can be easily compared with the gathered scanner measurements.

A.2 Model simplification

The original model of the factory, even after conversion to supported file formats such as Collada or Wavefront file format, was not suitable for experiments in the robotic simulator because the model was too detailed and thus too memory-demanding. Therefore, it was

necessary to simplify it.

In the beginning, the model and its tree structure were analyzed in .dae file format. Since the fences and doors were in one tree branch away from other objects, they were removed before the analysis continued.

The goal of the analysis was to separate the objects based on semantics (e.g., individual manipulators, lifters, and closets) connected or constructed from multiple pieces with different levels of detail and different dimensions (e.g., a manufacturer's logo versus a large concrete pedestal under a robot). The total number of these pieces (i.e., leaves of the model tree) was over 10^5. After the automatic and manual analysis, the model tree was pruned in different levels to obtain 254 objects (leaves of the pruned model tree).

There were 43 manipulators, 8 car body holders, 8 type tag readers, 4 lifters, 3 filtration stations, 2 conveyors, and other objects. Since it was unnecessary to have multiple objects of the same type, a subset of them was selected to create a set of **template objects** with 37 models (8 of them were different versions of manipulators). The rest of the objects were just rotated, translated, and sometimes slightly modified versions of the unique ones.

This small set of template objects was converted from Collada file format to Wavefront object file in Blender [47]. The converted objects were manually checked in MeshLab [48], and useless details like a warning text in Fig. A.1, were removed. Then the objects were cleaned by removing duplicate vertices, duplicate faces, zero area faces, and unreferenced vertices in MeshLab [48]. The cleaned objects were then simplified by merging close vertices. After cleaning and simplification, the saved objects were small enough for the robotic simulator, having about a tenth of the original file size, although the difference between the original model and its simplified version is not visible, as shown in Fig. A.2.

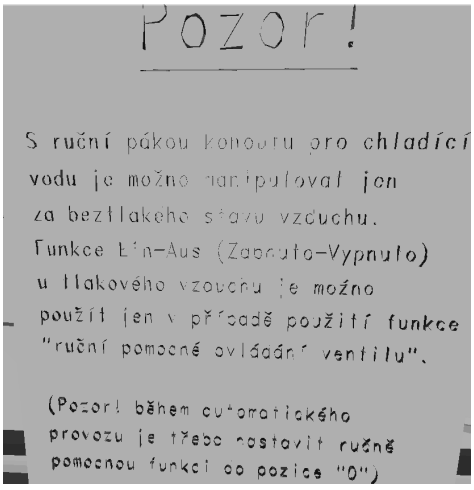


Figure A.1: Warning text label on one of the objects.

A.3 Model generation from configuration file

As the template objects were taken directly from the model tree with their locations, all their vertices were shifted, which caused unwanted offsets. Therefore, the template objects were

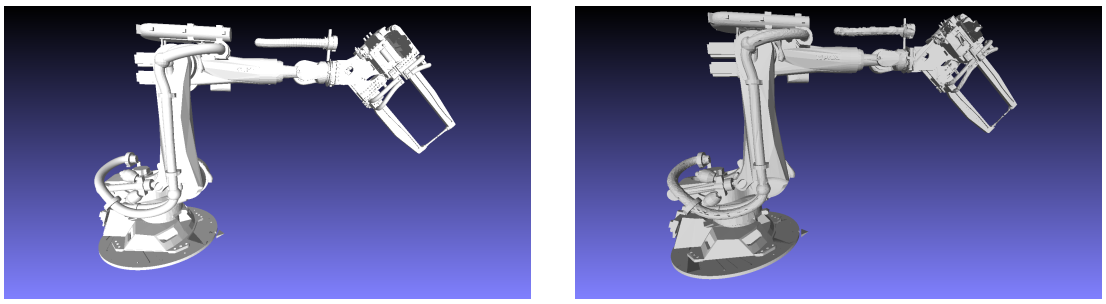


Figure A.2: Original model (left) and the simplified one (right) of the KUKA manipulator.

moved to their centroids in the x-y plane to remove these offsets. Moreover, they were aligned in the x-y plane, with the bottom starting at z=0. As a result, the template objects lie on the floor, and their centroids in the x-y plane are in (0,0).

Since the measured data are stored in point clouds, the template objects were sampled to create reference point clouds comparable to the measured data. Object surfaces were uniformly sampled with a density of 1 point per cm^2. The Ignition Gazebo model of the environment, its corresponding sampled version (i.e., point cloud), and the voxelized version can be seen in Fig. A.4.

As the final step, a configuration file for the factory model generation was created. Each row contains the object name, its position in 3D (usually z=0), its orientation around the axes (only the orientation around z-axis is not equal to zero), and the name of the template object from which the object was derived. An example snippet of the configuration file can be seen in Fig. A.3. The configuration file defines the factory model completely, and thus can be used to create different models in different file formats for different parts of the testing pipeline, such as, Gazebo model for the Ignition Gazebo, polygonal mesh for visualization, the reference point cloud for comparison with measured data, the binvox model and its octree variant for OctoMap library, which is used for evaluation. Since the file can be easily modified, it is possible to create many environments with different objects or different poses of objects.

The model preprocessing pipeline was designed to be easily used in different scenarios with various applications. New environments can be created by extending the set of template objects.

Node name	X [m]	Y [m]	Z [m]	RX [rad]	RY [rad]	RZ [rad]	Filename
kuka_type2_6	13.776	8.4221	0	0	0	1.5708	kuka_type2
kuka_type2_7	-8.945	12.6574	0	0	0	-1.5708	kuka_type2
kuka_type2_8	-8.945	8.5274	0	0	0	1.5708	kuka_type2
kuka_type3	11.5638	8.9201	0	0	0	1.5678	kuka_type3
kuka_type3_1	5.5788	12.6981	0	0	0	-1.5528	kuka_type3
kuka_type4	10.2246	12.8751	0	0	0	-1.5828	kuka_type4
kuka_type4_1	10.4534	8.5385	0	0	0	1.5588	kuka_type4
kuka_type5	16.2972	12.9563	0	0	0	-1.5688	kuka_type5
kuka_type5_1	16.315	8.4544	0	0	0	1.5758	kuka_type5
kuka_type5_2	7.7502	13.1519	0	0	0	-1.5738	kuka_type5
kuka_type5_3	7.7852	8.2891	0	0	0	1.5388	kuka_type5
kuka_type6	-9.045	-1.4675	0	0	0	-1.5708	kuka_type6
kuka_type6_1	-8.8134	-5.6674	0	0	0	0.7854	kuka_type6
kuka_type6_2	3.4465	12.9949	0	0	0	-1.5758	kuka_type6
kuka_type6_3	3.4495	8.3918	0	0	0	1.5738	kuka_type6

Figure A.3: Example snip of the configuration file with the header.

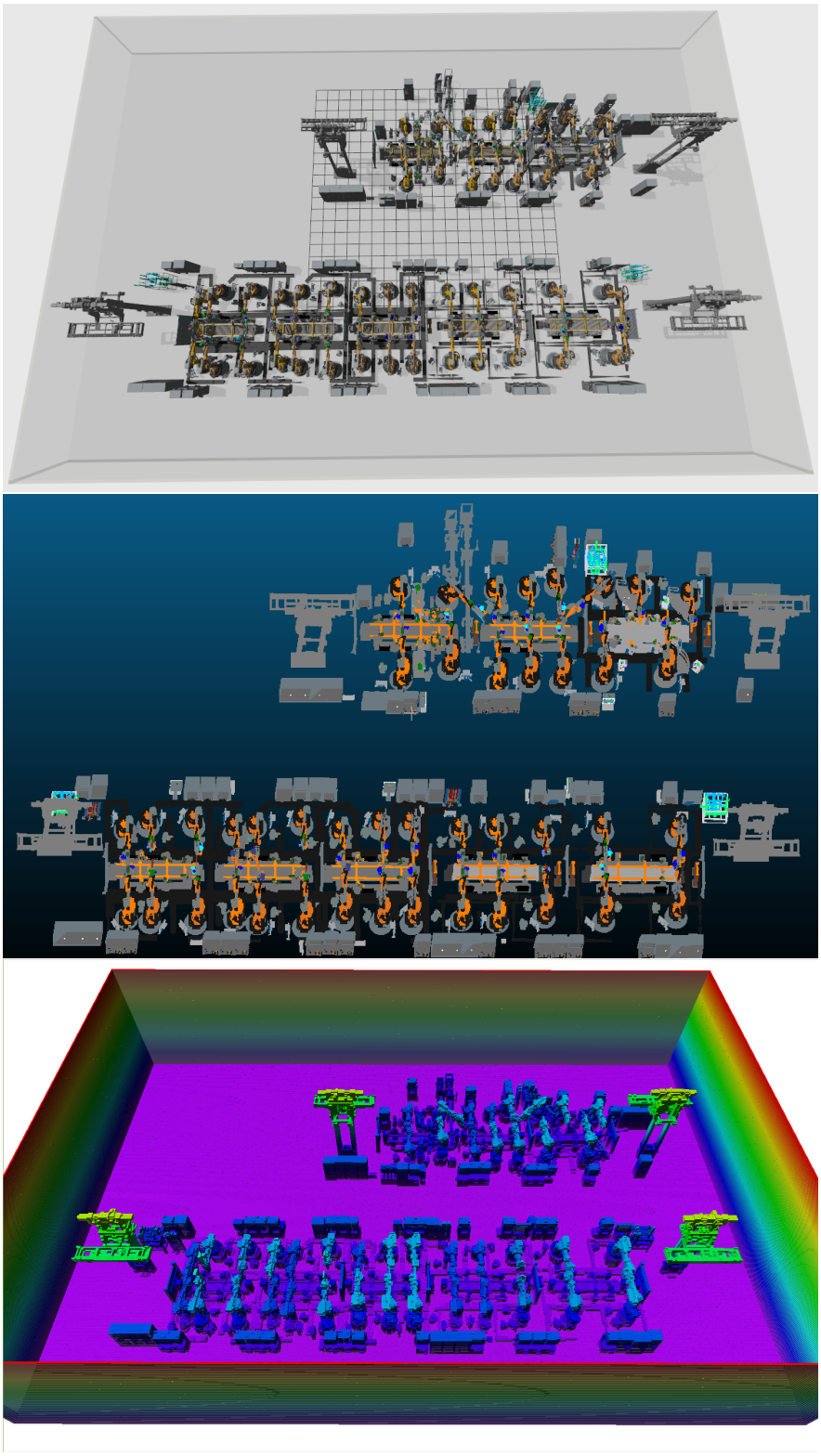


Figure A.4: Ignition Gazebo model example (top), its corresponding sampled version (middle), and the voxelized version (bottom).

MASTER'S THESIS ASSIGNMENT

I. Personal and study details

Student's name: **Rozlivek Jakub** Personal ID number: **466263**

Faculty / Institute: **Faculty of Electrical Engineering**

Department / Institute: **Department of Cybernetics**

Study program: **Cybernetics and Robotics**

Branch of study: **Cybernetics and Robotics**

II. Master's thesis details

Master's thesis title in English:

Automating 3D Scanning of Factory Hall by a Mobile Robot

Master's thesis title in Czech:

Automatizace 3D skenování tovární haly pomocí mobilního robotu

Guidelines:

The work will explore the possibility of using an autonomous robot to automate 3D measurement-scanning in factory halls. The aim is to automate 3D measurement and checking the position and orientation of individual robot manipulators and robotic cells during the construction or reconstruction of the factory arrangement. The work is based on the experience of the CTU-CRAS-NORLAB robotic team in the DARPA SubTerranean Challenge, where exploration optimizes the coverage of an unknown underground terrain. This work will propose a new exploration algorithm which will optimize the completeness of 3D data needed for the installation check. Any prior information about the scene, e.g. a CAD plan, should be exploited if suitable. The proposed solution will be modeled and tested in a physical robotic simulator. A prototype of a possible solution will be assembled from available robotic and sensory equipment. The prototype will be tested in a real factory hall environment. The output of the work will be a study of technical feasibility and the new exploration algorithm.

Bibliography / sources:

[1] P. S. Blaer and P. K. Allen. Data acquisition and view planning for 3-D modeling tasks. In IEEE International Conference on Intelligent Robots and Systems, 2007.
[2] N. Palomeras, N. Hurtos, E. Vidal, and M. Carreras. Autonomous exploration of complex underwater environments using a probabilistic next-best-view planner, IEEE Robotics and Automation Letters, vol. 4, no. 2, 2019.
[3] P. Kim, J. Chen, and Y. K. Cho. SLAM-driven robotic mapping and registration of 3D point clouds. Automation in Construction, vol. 89, 2018.
[4] L. M. González-de Santos, L. Díaz-Vilariño, J. Balado, J. Martínez-Sánchez, H. González-Jorge, and A. Sánchez-Rodríguez. Autonomous point cloud acquisition of unknown indoor scenes, ISPRS International Journal of GeoInformation, vol. 7, no. 7, 2018.
[5] Z. Meng, H. Qin, Z. Chen, X. Chen, H. Sun, F. Lin, and M. H. Ang, "A two-stage optimized next-view plan- ning framework for 3-D unknown environment exploration, and structural reconstruction," IEEE Robotics and Automation Letters, vol. 2, no. 3, 2017.
[6] CTU-CRAS-NORLAB tým pro soutěž DARPA SubT Challenge, http://robotics.fel.cvut.cz/cras/darpa-subt/

Name and workplace of master's thesis supervisor:

prof. Ing. Tomáš Svoboda, Ph.D., Vision for Robotics and Autonomous Systems, FEE

Name and workplace of second master's thesis supervisor or consultant:

Date of master's thesis assignment: **24.06.2021** Deadline for master's thesis submission: **04.01.2022**

Assignment valid until: **19.02.2023**

prof. Ing. Tomáš Svoboda, Ph.D. Supervisor's signature	prof. Ing. Tomáš Svoboda, Ph.D. Head of department's signature	prof. Mgr. Petr Páta, Ph.D. Dean's signature

III. Assignment receipt

The student acknowledges that the master's thesis is an individual work. The student must produce his thesis without the assistance of others, with the exception of provided consultations. Within the master's thesis, the author must state the names of consultants and include a list of references.

.	
Date of assignment receipt	Student's signature