UNIVERSITÀ DI PARMA

*Dottorato di Ricerca in Tecnologie dell'Informazione*

*XXXI Ciclo*

# Terrestrial Laser Scanning as a Support to Design and Deployment of Automated Warehouses

Coordinatore:

*Chiar.mo Prof. Marco Locatelli*

Tutor:

*Chiar.mo Prof. Jacopo Aleotti*

Dottorando: *Mikhail Giorgini*

Anni 2015/2018

*Feel free,*
*even in two lines.*

# Abstract

The design of modern automated warehouses is a complex task that requires detailed and accurate models of industrial buildings. However, the traditional design workflow is primarily based on 2D floor plans that may contain outdated or even missing parts. The common approach to update existing 2D floor plans is to perform surveys based on sparse measures, taken by hand-held instruments, such as Laser Distance Meters, or Total Stations. Indeed, these instruments provide very accurate point-to-point measures, but the amount of data that can be acquired may not be sufficient. As a result, multiple surveys need to be performed as new requirements emerge during system development. Moreover, many unpredictable issues are usually discovered at the time of deployment, thus resulting in high costs and delays. Therefore, companies that operate in the warehouse automation business are starting to apply 3D Terrestrial Laser Scanning technology to overcome the limitations of traditional surveys. This brings new research challenges, never addressed in industrial environments before.

A first contribution of this dissertation is the proposal of a novel workflow for the design of automated warehouses that improves the traditional development process by performing a 3D survey that combines a Terrestrial Laser Scanner and a Total Station. Automated warehouses include Autonomous Guided Vehicles (AGVs) that move along predefined paths, as well as fixed robot workcells. The workflow covers every phases from data collection to data processing, and exploitation. The proposed workflow can handle billions of points, acquired from thousands of scan stations, and it consists of several automatic and semi-automatic steps. The first step of the workflow is to perform a 3D survey by following a procedure that achieves a good trade-off between survey time, accuracy, and level of detail of the acquired point cloud. Then, high level information about the environment are extracted, exploiting innovative algorithms for large scale point cloud processing.

In particular, novel approaches for ground segmentation, floor plan generation and real-time AGV collision detection are presented. The proposed approach for ground segmentation does not assume the presence of a dominant plane and it scales linearly with the number of scans. The algorithm for floor plan generation is based on the segmented ground, so that no assumptions are made about the maximum expected slope. Moreover, walls are neither required to be planar nor having orthogonal intersections, like in previous works. AGV paths can, therefore, be defined based on the generated floor plan and a real-time collision detection algorithm is proposed to verify their feasibility. Virtual Reality is also supported to provide immersive visualization. Finally, a novel approach for scan position optimization is investigated that exploits a realistic sensor model that simulates a number of fixed parameters

having a strong influence on the laser measurements, like laser height from the ground, resolution, sensor range and angle of incidence of beams on both walls and ground.

While some parts of the proposed workflow have been developed to solve specific problems of the warehouse automation industry, most of the developed algorithms, such as automatic ground segmentation, floor plan generation and scan position optimization can be applied in any indoor environment. All the solutions developed in this thesis have been fully integrated with existing softwares to speed up the deployment phase. Experiments have shown that the proposed workflow drastically speeds up development and deployment of system installations.

# Contents

# Chapter 1

# Introduction

This chapter introduces the key research contributions of this work, along with the adopted technology. First, the chapter starts with a presentation of the industrial issues that motivated this work (Section 1.1). Then, Section 1.2 introduces the measuring instruments, pointing out their strengths and presenting the strategies that have been adopted to overcome their weaknesses. Finally, an overview of the following chapters, highlighting the scientific contribution and the novelty of the proposed solutions, is presented in Section 1.3.

## 1.1 Motivation

This thesis focuses on the design of automated warehouses. Automated warehouses include Autonomous Guided Vehicles (AGVs) that move along predefined paths, as well as fixed robot workcells. Modern design of automated warehouses is primarily based on 2D layouts of the customer buildings. Such layouts are very inaccurate due to missing or outdated parts. For example, clearances (i.e. machines, racks), but also structural elements (such as columns and walls), can be misplaced by metres in many cases. When it comes to warehouse retrofitting the ability to integrate proprietary solutions on existing and operating plants is crucial. In these scenarios accuracy requirements are higher than the accuracy of most available layouts. As an example,

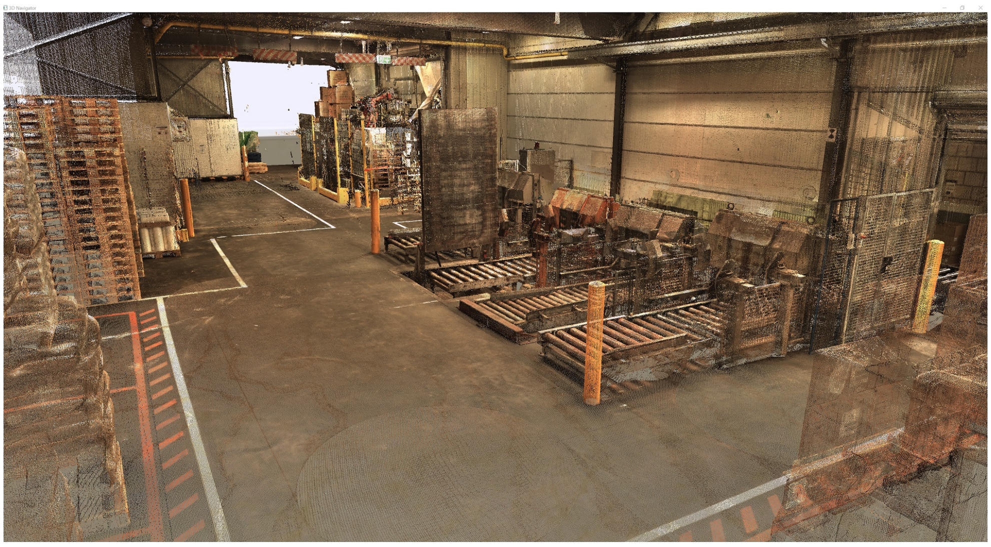


Figure 1.1: Example of point cloud acquired with a Terrestrial Laser Scanner.

for robot workcell placement an accuracy of about 10 cm may be required, while for AGV navigation and operations the maximum acceptable error is 2 cm. Hence, Terrestrial Laser Scanning technology may help to acquire an as-is model of the plant. Unfortunately, this technology provides huge amount of 3D raw data, in the form of point clouds (Fig. 1.1). The development of appropriate procedures and algorithms that can extract high level information becomes, therefore, crucial. Furthermore, having a reliable as-is model of the customer plant allows different development steps to be anticipated.

The as-is model of the plant can be exploited for accurate robot workcell placement. In fact, checking the feasibility of the installation at an early stage allows modification of the machine design in advance.

In addition, AGVs move along specified paths and perform operations (such as pallet loading and unloading) on predefined operation points. Planar laser scanners at the bottom of a vehicle are used for safety reasons, using predefined safety zones, so that if a laser sensor detects an obstacle within a safety zone it immediately triggers the vehicle to stop. Maximum speed is also statically determined by environment conditions and vehicle kinematics and it is strictly entangled with the size of safety zones. These settings are traditionally specified months before the system installation, but they are refined only when AGVs are deployed in the real environment. The refinement of these settings is a very time consuming task and it is currently performed during the deployment of the installation, that is the most delicate phase of a project. Delays in this phase have, in fact, a huge impact on the customer production activities, forcing the customer to stop large areas of the plant. Thus, having a reliable and detailed as-is model of the plant allows the determination of operation points, paths, maximum speed and safety zones long before the system installation, with negligible impact on customer production and distribution.

This thesis has been funded by Elettric80 SpA. Elettric80 was established in the 1980s in Viano, in the province of Reggio Emilia (Italy). The company is specialized in the implementation of integrated and automated solutions, flexible and modular, designed for high-volume consumer products manufacturers, mainly in the food, beverage and tissue industries, as well as in diversified sectors, like ceramics and plastics. Elettric80 provides solutions which allow planning and control of production, storage, and shipping activities, with a significant increase in factory efficiency, and ensuring the total traceability of handled products. The main systems, designed and customized according to the client's application needs, are: palletizing robots (Fig. 1.2), a wide range of Autonomous Guided Vehicles (Fig. 1.3), high-speed stretch wrappers (Silkworm), depalletizing and pallet control systems, robotic labelers, AS/RS warehouses (Stacker Crane Solutions) and Smart Store (Multilevel Solution), picking and repacking systems. The whole process is managed by a single software platform developed by Elettric80, named SM.I.LE80 (Smart Integrated Logistics), which ensures a direct link between systems and production processes, and the optimal and effective management of all internal and external plant operations: from incoming raw materials to complete warehousing and shipping management. Currently, Elettric80 has installed more than 1,700 robotic systems and 4,300 Autonomous Guided Vehicles worldwide. Beyond its headquarters in Viano, Elettric80 has set up branches in Australia, Brazil, Chile, the United Arab Emirates, France, Great Britain, Mexico, Poland, Russia, Sweden, and the USA.

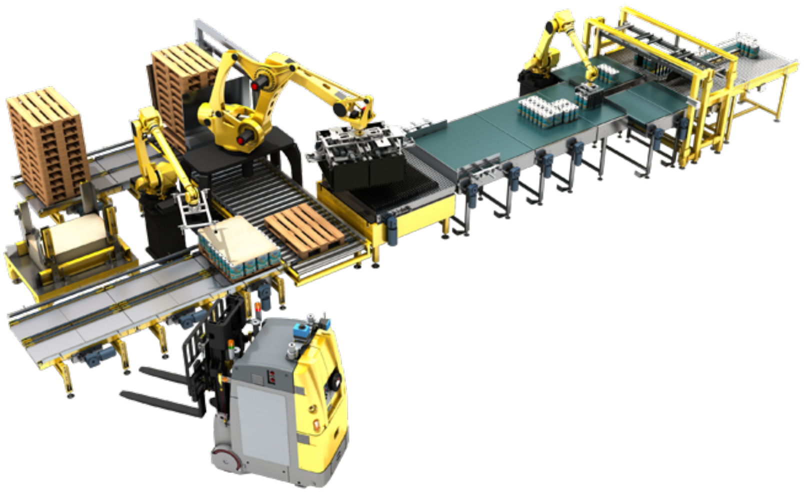


Figure 1.2: The DRAGON Tissue palletizing cell features Fanuc robots to form individual packages into layers and a larger Fanuc robot fitted with a patented "forks gripper" to place the layers on a pallet.

Figure 1.3: Autonomous Guided Vehicles provide a fast and flexible solution for pallet storage, as they reduce time, cost, error and risk, compared to human operators.

Figure 1.4: Leica ScanStation P30.

## 1.2 Adopted sensing technologies

Terrestrial Laser Scanners (TLSs) are becoming a standard choice to acquire accurate and highly dense data of indoor environments. Many companies that actively use BIM (Building Information Modeling) in their workflow are in fact investing in TLSs as they provide an accurate and affordable solution.

A Terrestrial Laser Scanner is a ground-based instrument that is able to acquire accurate 3D point clouds of environments by a range-based laser technology. 3D measures are reconstructed from two angles (azimuth and elevation), coming respectively from horizontal and vertical encoders, and a distance measure, provided by an EDM (Electronic Distance Measurement).

Following the classification proposed in [27], this work will focus on medium-range static TLSs. Medium-range instruments provide the best trade-off between maximum range and accuracy when it comes to indoor measurement of large spaces, such as airports, public monuments, industrial warehouse and so on. In this thesis, the TLS adopted for data acquisition is a Leica Scanstation P30 (Fig. 1.4) mounted on a tripod. This high-end sensor features an angular accuracy of 8”, a beam divergence below 0.23 mrad and a position accuracy of 3 mm at 50 metres. It provides a scan

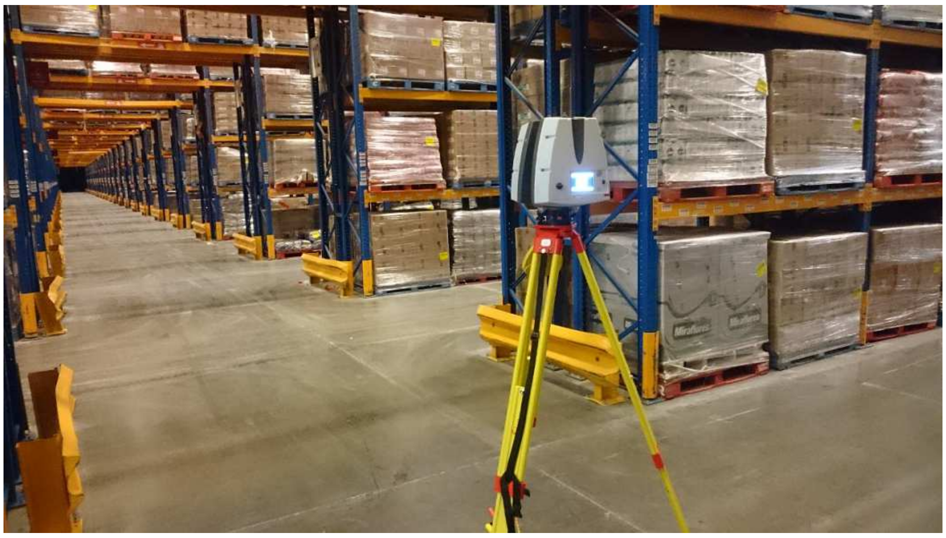


Figure 1.5: Terrestrial Laser Scanning survey in a warehouse using a Leica ScanStation P30 mounted on a tripod.

rate of 1 million points per second, and a maximum range of 120 m, with a field of view of $360° \times 290°$. The TLS needs to be approximately leveled in each scan station to reach the working range of the built-in dual-axis compensator (based on a liquid sensor) used for real-time measurement correction. Adopting a TLS-based approach, instead of other techniques like range sensors mounted on mobile platforms, provides higher accuracy. Other works, such as [5] and [11], adopted stop-and-go techniques to avoid mobile platforms problems, such as poor odometry information, parallax and vibrations. In this work, TLS surveys are performed from multiple scan stations (Fig. 1.5), that are later registered together.

Point cloud registration is the process of transforming point clouds, acquired from different scanner positions, to the same reference frame, named Uniform Coordinate System (UCS). This may be achieved with two main techniques, based on artificial targets (Fig. 1.6) or point cloud comparisons. In this thesis, a mix of the two techniques has been explored, to obtain a good trade-off between survey-time, accuracy, and level of detail of the reconstructed environment. Despite the fact that the P30 can work in

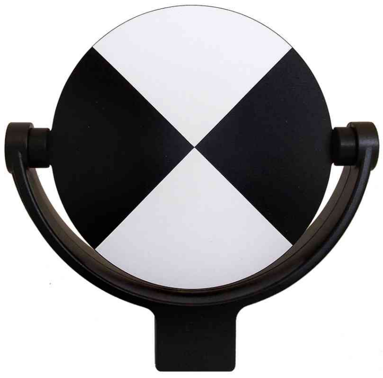


Figure 1.6: Artificial B/W Target for Leica ScanStation P30.

topographic mode (i.e. it can work with targets), the use of this instrument alone for large spaces is not recommended. In fact, the limited maximum range (120 m), combined with a low angular accuracy (8") cannot keep the global error under the requirements (2 cm for AGV navigation and operations). Furthermore, the procedure for target acquisition on TLSs is very slow. Hence, it has been decided to adopt a technique that combines the TLS with a Total Station for target measurements.

The Total Station is an electronic instrument that combines a theodolite, for angular measurements (both horizontal and vertical), with an Electronic Distance Measurement (EDM), for slope distance. Total Stations provide the capability for 3D point computation and new pose estimation, based on triangulation algorithms, thanks to on-board software. Currently, Total Stations are the fastest and most accurate instruments for single-measure collection.

The Total Station adopted for this work is a Leica Nova TS60i 0.5" (Fig. 1.7). This high-end instrument features an angular accuracy of 0.5", a maximum range of 3500 m and an accuracy of 0.6 mm + 1 ppm on prisms (Fig. 1.8). This instrument can work both in manual and auto-aim mode. Auto-aim mode requires the use of prisms, whose center can be automatically detected after a rough manual alignment, or in a fully automatic way with the use of the *Power Search* feature, that performs a fast 360° search.

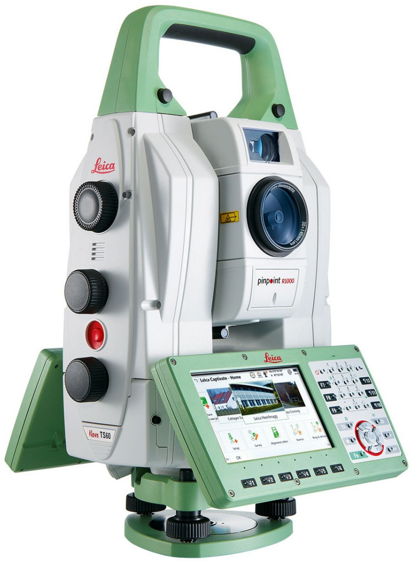


Figure 1.7: Leica Nova TS60i Total Station 0.5".

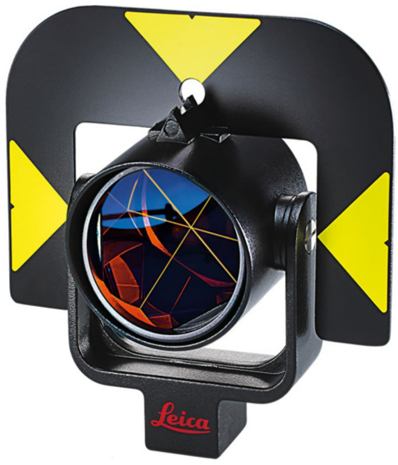


Figure 1.8: Circular prism with holder. It provides a centering accuracy of 1mm and a maximum range of 3500 m.

## 1.3 Thesis contribution and outline

In this section the scientific contribution of this thesis and its outline are described.

First, in Chapter 2, a novel workflow is presented that overcomes traditional 2D survey limitations. The benefits with respect to the traditional workflow are also discussed. Moreover, an overview of the proposed workflow steps is provided. The main scientific contributions of the thesis are then discussed separately in the following chapters.

A novel approach for ground segmentation in a registered large-scale point cloud acquired from a Terrestrial Laser Scanning survey is presented in Chapter 3. The proposed method is robust to clutter and it is capable of extracting the ground even in presence of dense planar regions that do not belong to the ground. The segmented ground is important as it provides a reference for subsequent algorithms and an accurate analysis of the ground status, that is crucial for safe AGV navigation.

Then, a novel approach for floor plan generation, particularly suited for complex buildings like industrial premises, is presented in Chapter 4. The main novelty of this approach is that no assumptions are made about the presence of a flat ground nor a flat ceiling. Moreover, no assumptions are made about the maximum expected slope of the ground, as it exploits the ground segmentation. Furthermore, no hypotheses are also made about the planarity and orthogonality of walls. Cluttered areas are included in the generated floor plan, so that the floor plan can be exploited as a map for AGV path design.

A novel approach for AGV path validation that supports Virtual Reality (VR) visualization and real-time collision checking with large scale point clouds is presented in Chapter 5. The proposed approach has been integrated with the Elettric80 software, so that technicians can verify the feasibility of vehicle paths directly in the CAD application in which paths are developed.

Finally, a novel formulation of the optimal sensor placement problem that includes realistic constraints is presented in Chapter 6. A Terrestrial Laser Scanning survey is, in fact, usually performed without any guarantee of optimality, based on the experience of technicians who determine the scan station positions. The problem is solved as a modified version of the *Set Cover Problem*, as it requires an ordered solution to fulfil the overlap constraint that ensures the success of cloud-to-cloud registration algorithms (Section 2.3).

Conclusions, limitations and future work are finally presented in Chapter 7.

# Chapter 2

# Proposed Workflow for the Design of Automated Warehouses

This chapter presents the workflow adopted by Elettric80 that exploits the approaches and algorithms investigated in this thesis.

A comparison between the proposed and the traditional workflow is also presented and benefits of the former are discussed. The first section will provide a high level overview, while other sections will deepen each single step. The following chapters describe the developed algorithms and methods.

## 2.1 Overview

The traditional workflow in the automation business is primarily based on 2D layouts and single measures roughly taken by hand-held instruments, such as Laser Distance Meters. Only in few and rare cases layouts are integrated with measures taken using Total Stations. These instruments provide very accurate point-to-point measures, but the amount of acquired data may not be sufficient. As a result, multiple small surveys are performed as project requirements appear during the machine design. Stopping the machine development and taking multiple small surveys is very time consuming and prone to errors. Moreover, many unplanned issues are discovered at the time of

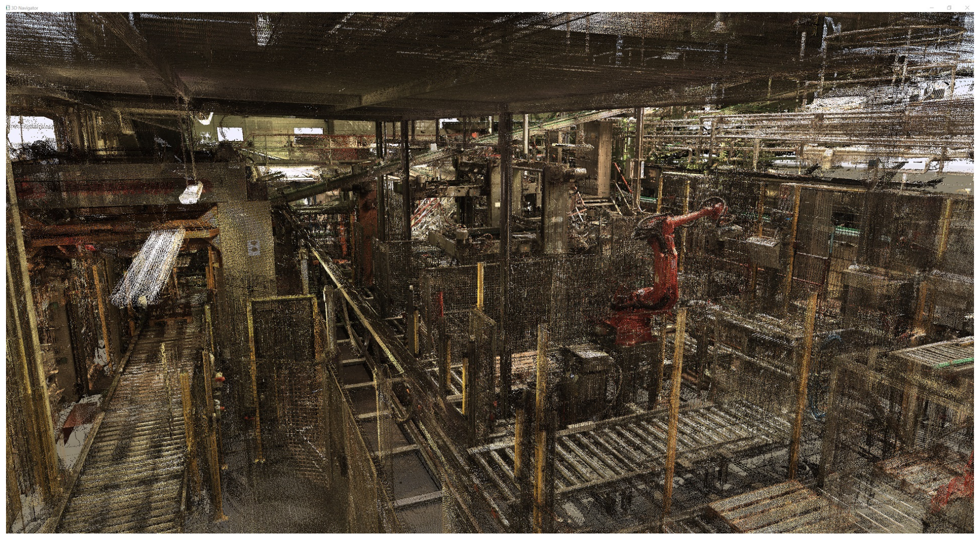


Figure 2.1: Example of point cloud of a complex building.

deployment, thus resulting in very high costs and delays. In addition, environments are typically complex in the automation business and the required level of integration between autonomous systems and the original building is high. As a solution, the whole 3D acquisition of the environment is performed as a preliminary step (Fig. 2.1). Workcell positions and AGV paths are then designed and verified on the 3D representation coming from the survey, introducing a virtual deployment phase. Fixes and corrections at this stage have in fact a low cost, compared to adjustments made during the deployment phase. Figure 2.2 shows the traditional workflow compared to the proposed one. It is evident that in the proposed workflow design loops are limited to the low cost / low risk tasks performed in the technical office. Furthermore, the adoption of the proposed workflow simplifies the transition from human-operated forklifts to Autonomous Guided Vehicles, as it avoids stopping the warehouse activities and emptying racks as in the past.

Figure 2.3 shows the data workflow, from the survey to the final plant validation, highlighting all the fundamental steps required to transform raw data into high level information.

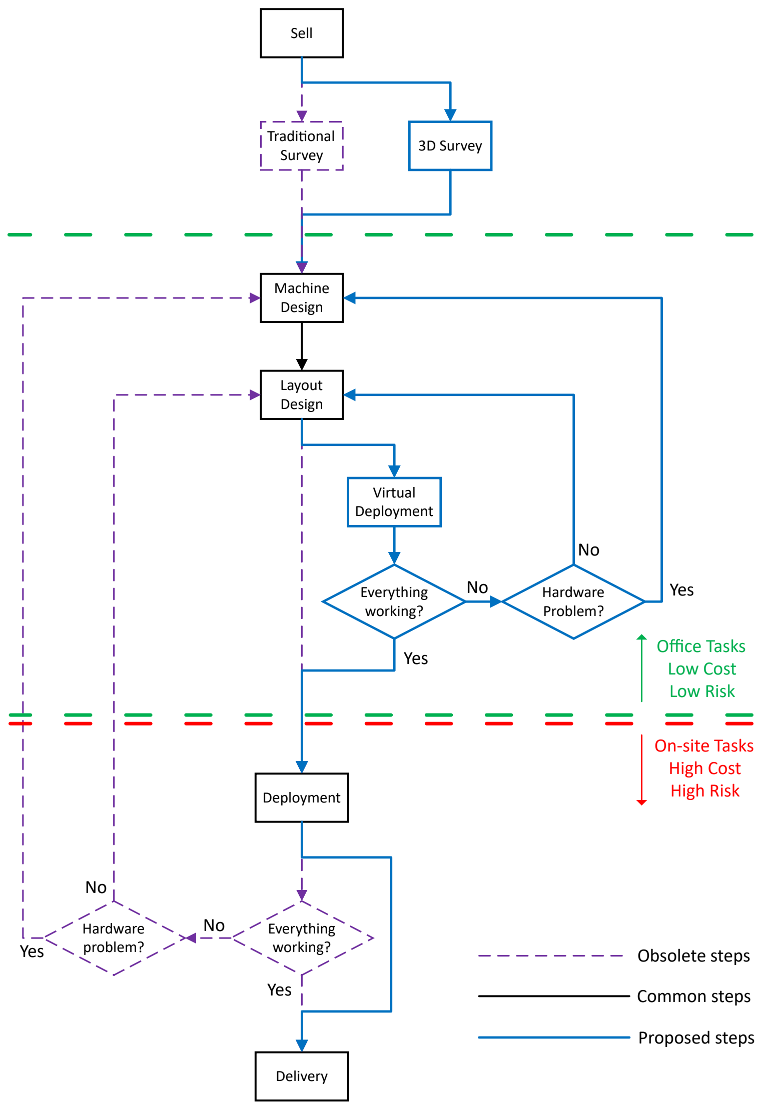


Figure 2.2: Traditional worflow (dashed lines) compared to proposed workflow (solid lines). Confining loops in the design process performed by the technical office, by introducing a virtual deployment step based on the 3D survey, significantly reduces risks and costs. In fact, problems discovered during on-site tasks have an higher impact in terms of cost and delays.

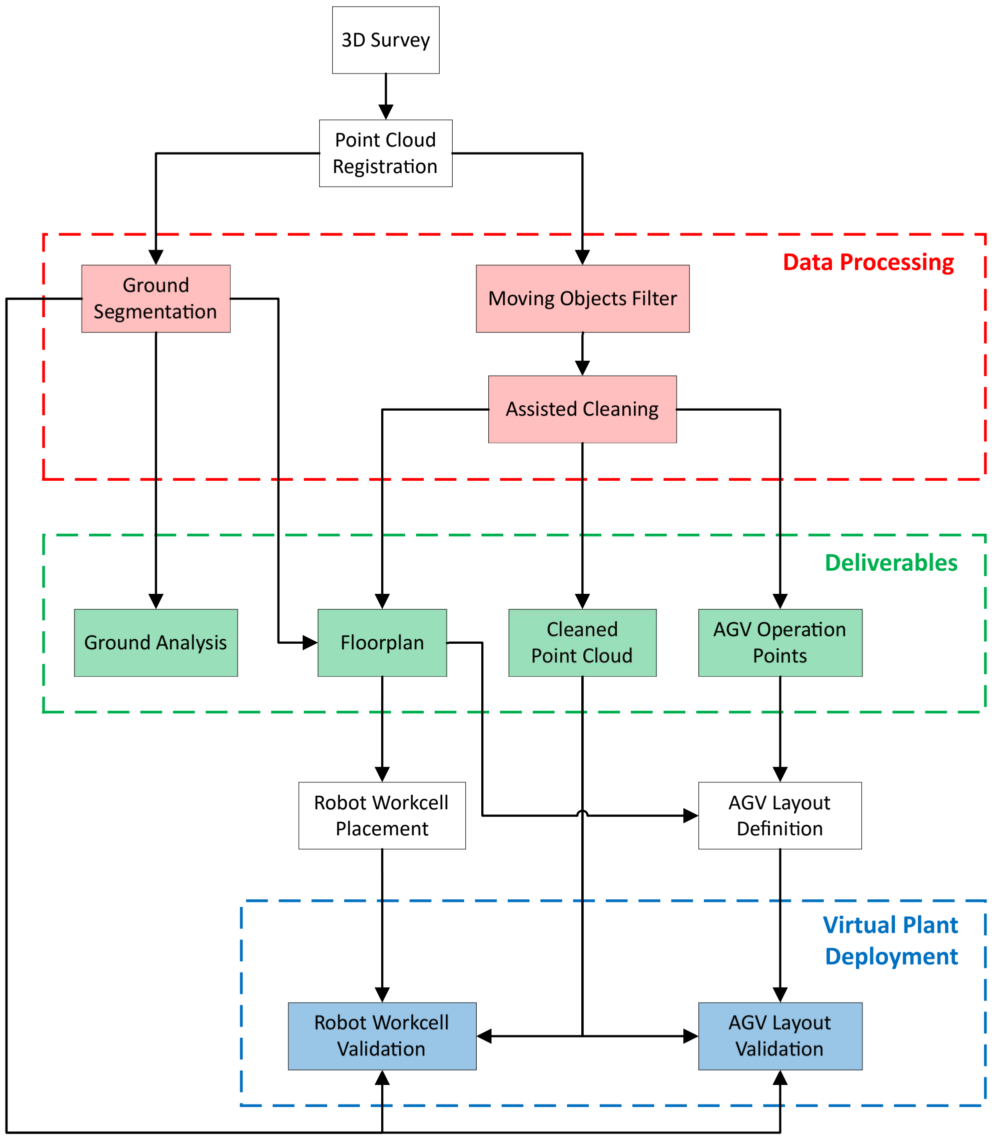


Figure 2.3: Data acquisition and elaboration workflow.

## 2.2 Survey

Survey is performed by combining the Terrestrial Laser Scanner with the Total Station to take advantage of both. In fact, low angular accuracy, low range as well as slow target acquisition of the TLS led to the decision of using a Total Station to reduce the global error and to speed-up the survey. In particular, the procedure adopted to perform a survey works as follows.

A limited number of 3D scans are performed in “topographic mode”, meaning that they are registered with artificial targets (acquired by the Total Station in advance), while the largest part of the survey is performed using “free stations”, meaning that they need to be registered with cloud-to-cloud techniques in post-processing. The first step of a survey is about deciding the locations of topographic scans. These locations have to maximize to visible area to facilitate registration of free stations with the corresponding topographic station. At the same time, the total number of topographic scans needs to be a small percentage of the total scans, to minimize the impact on the survey time. As an example, performing a free-station scan takes approximately 3 minutes, while topographic scans take more than 15 minutes. Once the locations of topographic scans have been decided, artificial targets are mounted to achieve adequate triangulation. Depending on the environment, different types of target supports are used (Fig. 2.4). Target supports are divided in two main categories: mobile and fixed. Mobile supports are removed after the survey is completed, while fixed supports are left for future use and are intended to be used as a permanent reference. Mobile supports can be magnetic or adhesive. Magnetic supports have a high cost and are typically removed at the end of a survey session, while adhesive supports are left overnight. Furthermore, some target supports are compatible with prisms (Fig. 1.8), which allows auto-aiming with the Total Station, thus speeding-up the survey and providing important benefits in terms of accuracy and confidence in the acquired measures. In fact, measures taken with manual aiming may be invalid due to human errors. Table 2.1 lists all the types of target supports. It is evident that no target has all the required characteristics, so a mixed use is required. At the time of writing, wall spigots are being evaluated. Wall spigots have the same characteristics of rack

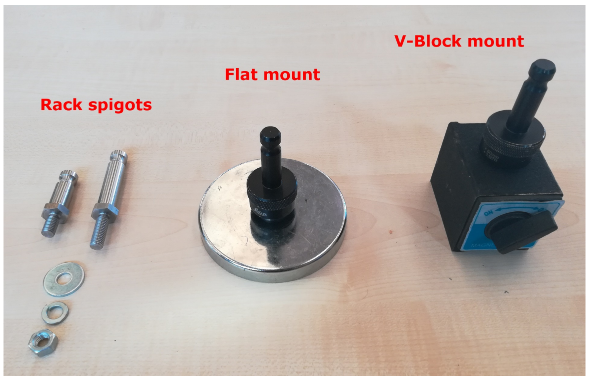


Figure 2.4: Supports for Leica 4.5" Tilt&Turn targets. Rack spigots (left) are essential in warehouses with racks, while magnetic mounts for flat (center) or cylindrical (right) surfaces have a wider applicability. The advantage of rack spigots lies both in their low cost (self-made by Elettric80) and their robustness to accidental collisions by operators. They are left on-site for multi-day surveys, but they do not represent a long term reference, because racks (on which they are mounted) suffer from structural deformation over time. See Table 2.1 for details.

spigots, but they are screwed directly on walls, thus providing long term reference.

Then, after the target supports have been mounted, they are mapped with the Total Station, using standard techniques for traditional surveys (polygonal chain, resection, backsight). Supports that are compatible with prism, such as rack spigot, flat and V-Block mounts are mapped exploiting the auto-aiming feature of the Leica Total Station TS60. Other targets are mapped by visually pointing their center with the camera or the monocular.

Once the targets are mapped, the 3D survey with the Terrestrial Laser Scanner starts. The surveyor proceeds in an ordered path along the plant to facilitate the subsequent registration phase, by alternating many fast free stations to some slow topographic stations when he reaches the strategic spots previously defined (in which targets have been placed). The adopted resolution is 1.25 mrad, that is a medium-low resolution for this category of instruments. In fact, the complexity of the environments in terms of clutter and occlusions impose a high number of scan stations in order to

Table 2.1: Target supports.

| **Name** | **Mount** | **TLS** | **Auto-Aim** | **Tilt&Turn** | **>1day** | **>1year** |
|---|---|---|---|---|---|---|
| Rack spigot | screw | ✓ | ✓ | ✓ | ✓ | |
| Reflective tape | adhesive | | ✓ | | ✓ | ✓ |
| Flat mount | magnetic | ✓ | ✓ | ✓ | | |
| V-Block mount | magnetic | ✓ | ✓ | ✓ | | |
| Forex 6¨ | adhesive | ✓ | | | ✓ | ✓ |
| Plastic sheet 6¨ | adhesive | ✓ | | | ✓ | |

have a detailed representation of the buildings. As a consequence, having a high number of scans at close distance makes the use of higher resolutions unnecessary.

Following chapters will refer to “TLS Survey” as the proposed procedure that combines the use of Total Station and Terrestrial Laser Scanner.

## 2.3 Registration

Registration is the act of aligning point clouds acquired from different scans positions, so that they are referred to the same coordinate system. Thanks to registration, the whole 3D representation can be created by combining all the scans. In the proposed workflow, registration is performed using “Leica Cyclone”, a proprietary software distributed by the TLS vendor. This is the only post-processing phase performed using a 3rd party software. The most used technique exploits 2D views of the scans (side-view and top-view). Scans are visually aligned from the top-view (Figs. 2.5 and 2.6), then the height is refined from the side-view (Figs. 2.7 and 2.8). The visual alignment provides a coarse guess, acting as a seed for an ICP algorithm, that is able to find the best cloud-to-cloud alignment. Information on convergence are then provided, leaving the user the final choice about the correctness of the alignment. The user iterates the alignment procedure on pairs of scans, trying to follow the same temporal order of the survey. The iteration of the procedure on pairs of adjacent scans highlights the importance of an ordered survey on-site.

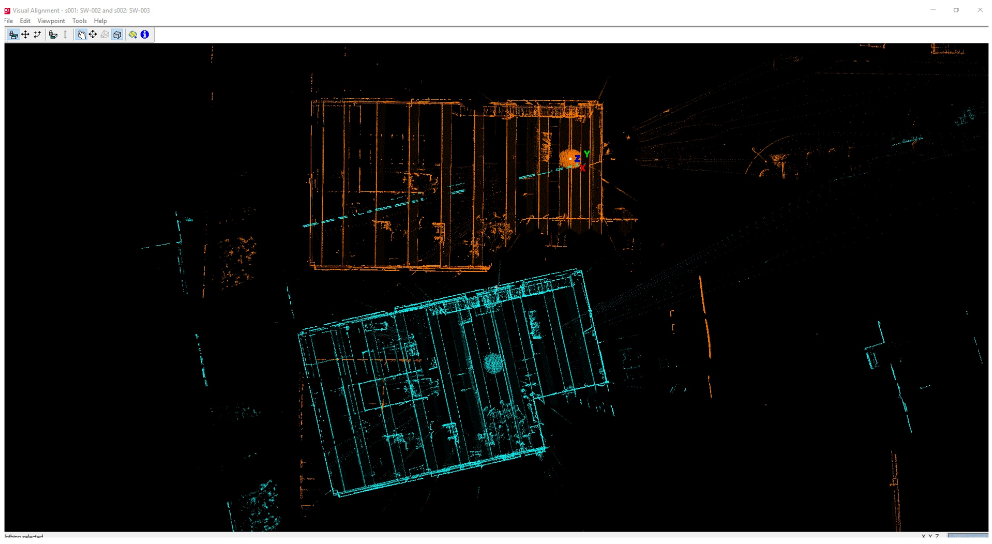

Visual Alignment - s001: SW-002 and s002: SW-003
File Edit Viewpoint Tools Help

Figure 2.5: Registration software. Top view before alignment.

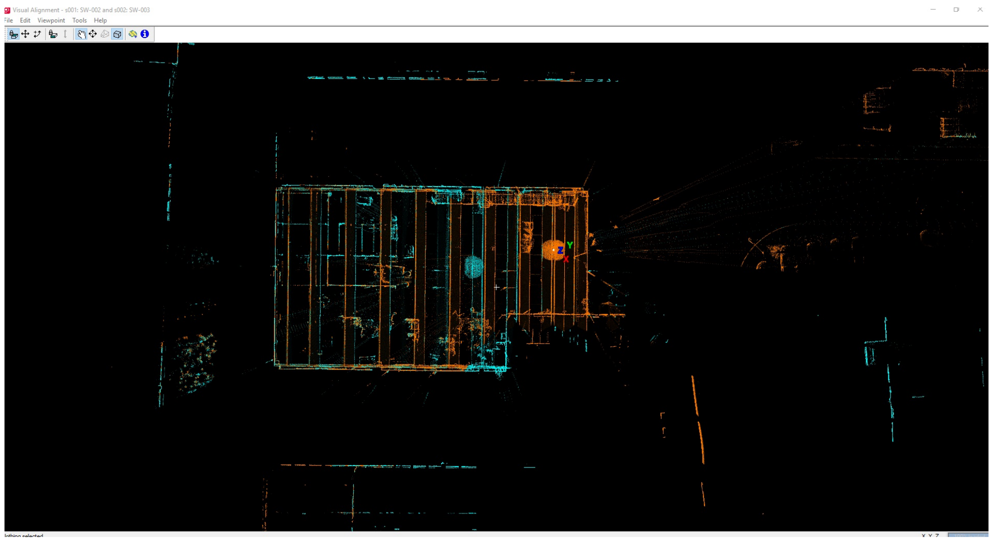

Visual Alignment - s001: SW-002 and s002: SW-003
File Edit Viewpoint Tools Help

Figure 2.6: Registration software. Top view after alignment.

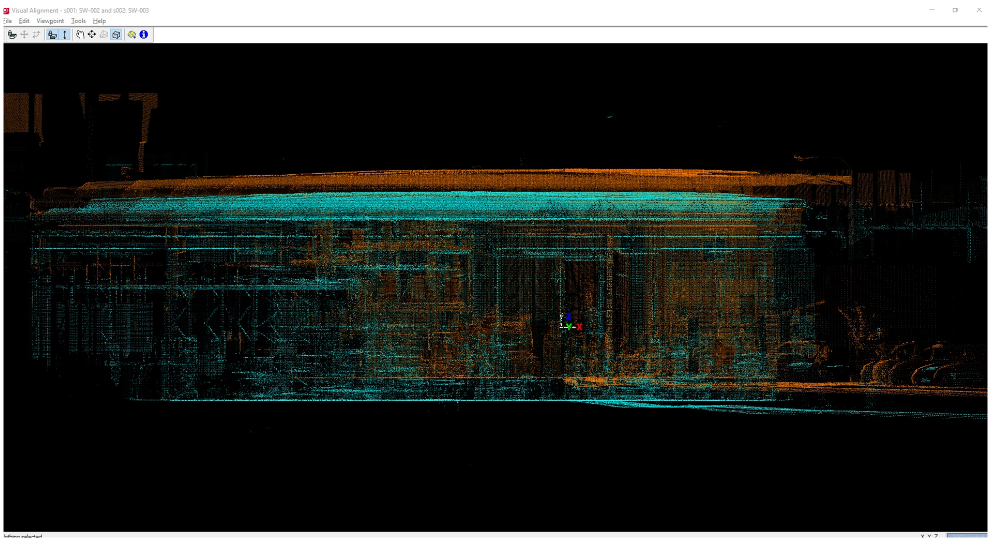

Visual Alignment - s001: SW-002 and s002: SW-003
File Edit Viewpoint Tools Help

Figure 2.7: Registration software. Side view before alignment.

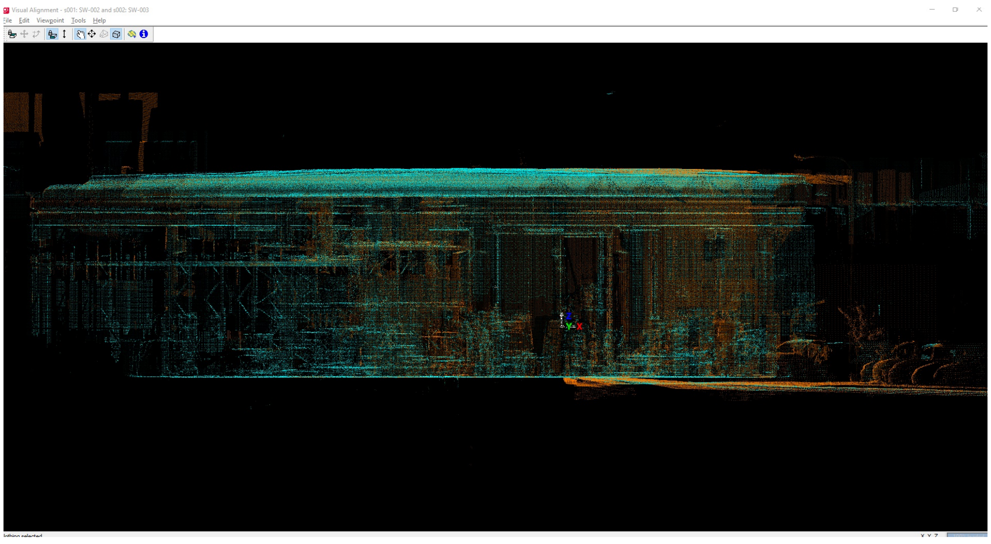
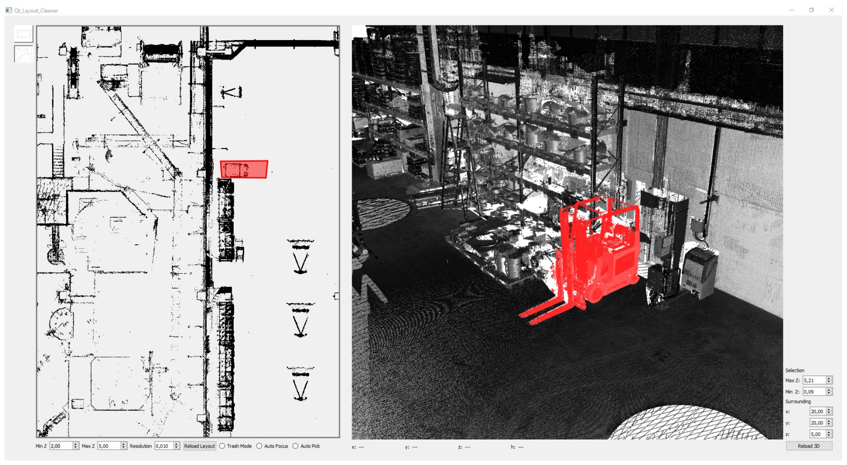

Visual Alignment - s001: SW-002 and s002: SW-003
File Edit Viewpoint Tools Help

Figure 2.8: Registration software. Side view after alignment.



Min Z 2,00 Max Z 5,00 Resolution 0,010 ○ Trash Mode ○ Auto Focus ○ Auto Pick

Selection
Max Z: 5,21
Min Z: 0,09
Surrounding
x: 20,00
y: 20,00
z: 5,00

Figure 2.9: Manual cleaning software. In this example a manual forklift has been manually selected for removal. The user interface features a 2D representation of the point cloud (left side) in which a polygonal or rectangular area can be selected. On the right side the height limits of the point cloud selection can be refined thanks to two spin boxes representing lower and upper bounds. Once the volume has been selected, point removal takes few milliseconds on a standard personal computer.

## 2.4 Data Processing

Data processing is performed thanks to the developed softwares and algorithms and consists of automatic and manual phases. The first automatic step performs ground segmentation [25]. Chapter 3 describes the proposed and adopted algorithm in detail. The output are the segmented ground points and a filled height map that can be used as a reference. Then, noisy points due to moving objects are automatically removed exploiting ray tracing techniques, similarly to [58] and [72]. Finally, data are manually processed to remove static clutter, using a software for assisted point cloud cleaning that combines the intuitiveness of a 2D representation with the completeness of a 3D representation for accurate point removal. Figure 2.9 shows the user interface of the software. A video of the software exploited to remove static clutter is also available.[1]

[1]`www.ce.unipr.it/~aleotti/ManualCleaning.mp4`

## 2.5 Deliverables

Once data have been automatically cleaned from moving objects and manually cleaned from static clutter, they are ready for high level information extraction. In this thesis deliverables are defined as the materials that are produced within the proposed workflow. In particular, deliverables include ready-to-use documents, reports and results that aggregates raw data into high level information. Deliverables are meant to be used by the technical office in the design phase, and to be shared with the customers to highlight critical issues or to increase the commercial appeal of the offer.

The most important deliverables are:

- Ground analysis
- Floor plan
- Cleaned point cloud
- AGV operation points

The ground analysis is meant to be shared with the customer, along with instructions on fixes needed to comply with the installation requirements. The floor plan is mainly for the technical office use, with the purpose of robot workcells placement and AGV layout definition. The cleaned point cloud is also meant for internal use for point-to-point measures and consultation. Under specific circumstances and after explicit request the cleaned point cloud can be shared with the customer. The set of the AGV operation points is the last deliverable coming from data processing and it is meant only for internal use and exploited for the design of AGV layouts.

Next subsections will present deliverables in detail with some real-case examples and highlighting the specific importance of each one.

### 2.5.1 Ground analysis

To obtain precise and repeatable operations, most of the Autonomous Guided Vehicles do not feature a suspension system. For this reason, small ground imperfections have a significant impact on AGV navigation and operations. In fact, during navigation,

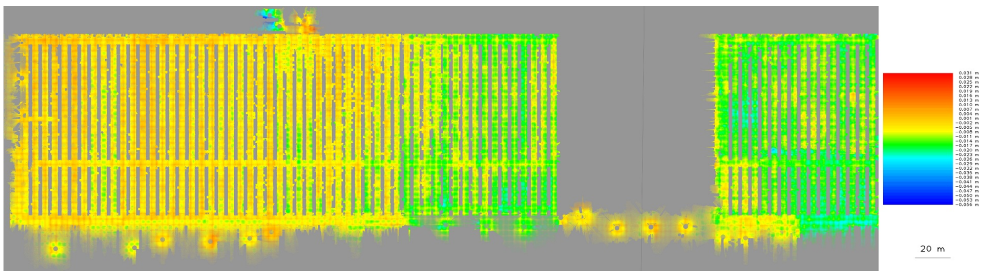


Figure 2.10: Height map example. Each pixel of the image corresponds to the local height of a square centimetre. This image is used in conjunction with the *Height-Deviation* map to inspect and identify ground problems that require fixes before the system installation.

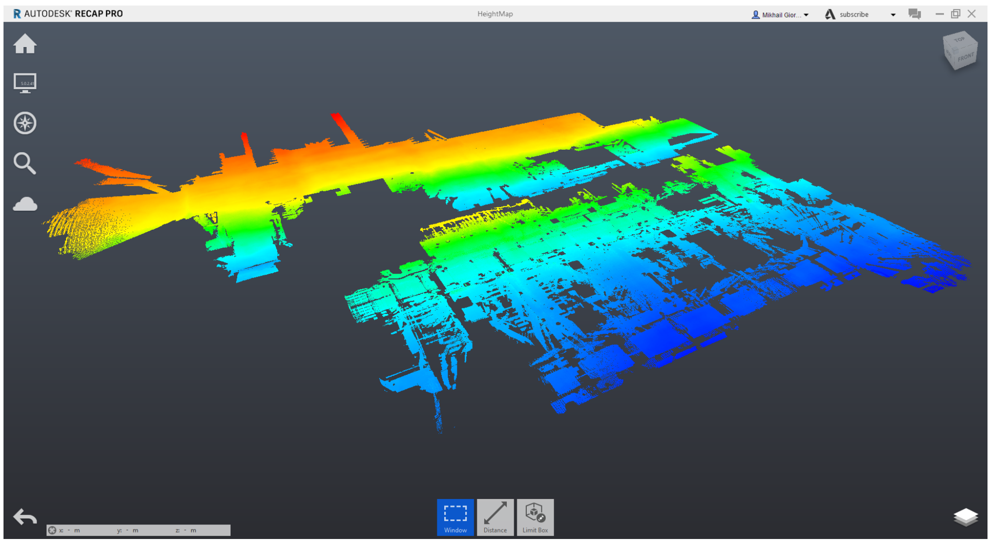


Figure 2.11: Example of the 3D point cloud of the segmented ground. The same cool-to-warm color scale of the height map is used to highlight elevation differences. The point cloud is exported using the E57 file format, that can be imported in Autodesk ReCap for inspection.

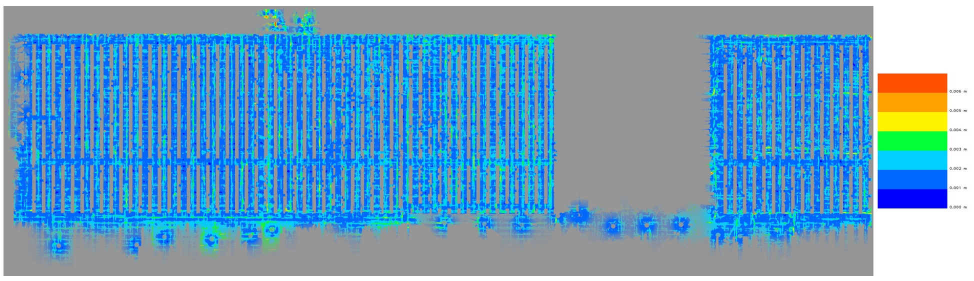


Figure 2.12: *Height-Deviation* map example. Each pixel of the image corresponds to the difference between the local height and the mean height of the surrounding square metre. The size of the considered surrounding is customizable depending on the application.

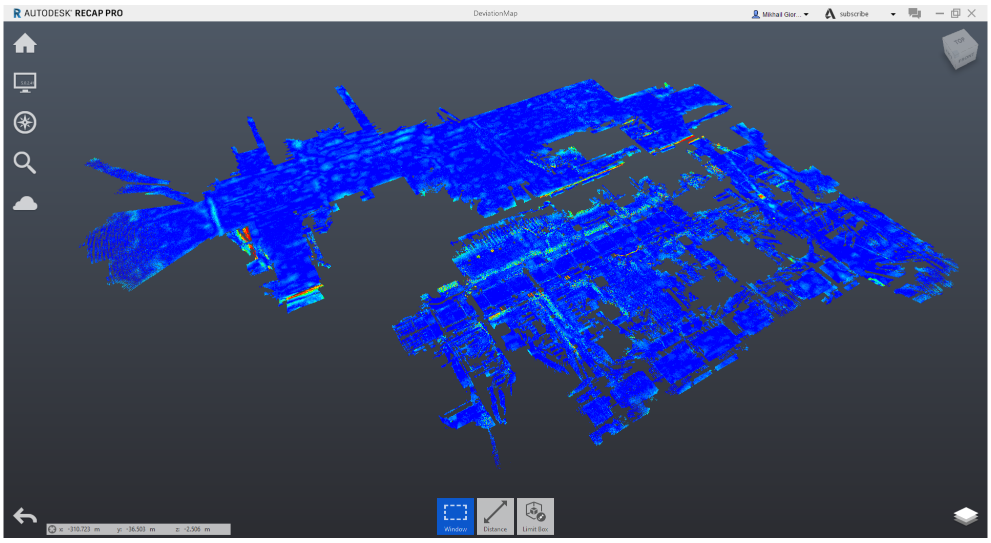


Figure 2.13: Example of the 3D point cloud of the segmented ground. The same cool-to-warm color scale of the *Height-Deviation* map is used to highlight areas that may be critical for AGV navigation.

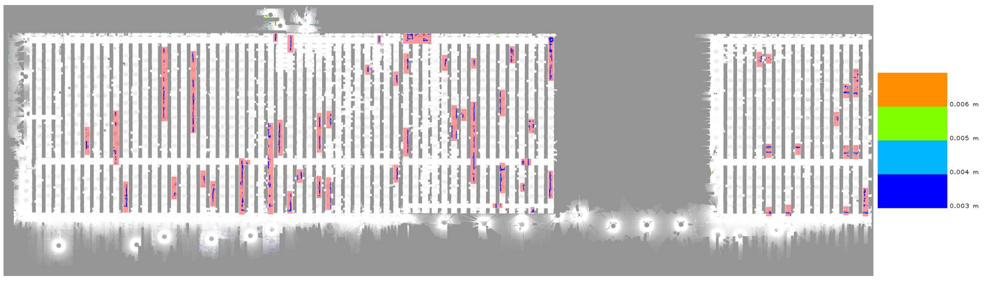


Figure 2.14: Ground analysis provided to the customer, based on the *Height-Deviation* map. Red rectangles highlight areas that require intervention, while white pixels represent areas that do not require any fixes.

fractures, holes and bumps may irreversibly damage different parts of the vehicle. Moreover, due to the fact that AGVs always travel on the same paths, they tend to aggravate ground problems. Furthermore, when performing operations in positions higher than 10 metres, ground height differences of few millimetres cause errors of centimetres in the final pallet configuration. In these conditions performing an operation results in wrong pallet forking and it may cause damages to structures, goods and to the vehicle itself. For this reasons, it is important to analyse the condition of the ground surface in advance to ask the customer to fix the imperfections before the system installation. Therefore, a novel algorithm is proposed to segment the ground in a registered large-scale point cloud. Chapter 3 presents the segmentation algorithm.

Ground analysis is delivered as color maps, in the form of images (.png) and point clouds. Point clouds are exported in the E57 format that is compatible with Autodesk ReCap, a commercial software for point cloud visualization that allows complex measures (point-to-point, point-to-surface, surface-to-surface) and annotations.

Three types of deliverable for ground analysis are provided:

- Height map (Figs. 2.10 and 2.11)
- *Height-Deviation* map (Figs. 2.12 and 2.13)
- Ground analysis (Fig. 2.14)

The pixel size and the point cloud resolution are customizable depending on the application (typically 1 cm). A legend on the right depicts the association between colors of the image and the corresponding pixel values.

### 2.5.2 Floor plan and cleaned point cloud

The floor plan is generated exploiting the ground segmentation, after automatic removal of noisy points belonging to moving objects and manual cleaning. An algorithm for automatic floor plan generation is proposed that does not assume the presence of a flat ground nor a flat ceiling. The algorithm makes no assumptions about the maximum allowed slope of the ground and walls are neither assumed to be planar nor having orthogonal intersections, in contrast to previous works. Moreover, the layout machines and objects below the height limit is included in the floor plan, thus providing reliable navigation maps for AGV path definition or interference sections for robot workcells placing. Chapter 4 presents the proposed algorithm. Generated floor plans support Autodesk AutoCAD through the DXF format. Figures 2.15, 2.16, 2.17 and 2.18 show examples of generated floor plans. A video of the software exploited to generate floor plans is also available.[2]

The full cleaned point cloud is also exported using the E57 format to support Autodesk ReCap. Figures 2.19, 2.20, 2.21 and 2.22 show examples of exported point clouds. A video of the software exploited to export cleaned point clouds is also available.[3]

### 2.5.3 AGV operation points

Operation points are the (x,y) coordinates that define the spots in which Autonomous Guided Vehicles perform operations, such as pallet pick up and drop, truck loading and unloading, battery recharge, and so on. Traditionally, operation points were identified at the time of the system installation by manually bringing the AGV in the operation position. Moving an AGV into all the operation points was a very time consuming

[2]www.ce.unipr.it/~aleotti/FloorplanGeneration.mp4

[3]www.ce.unipr.it/~aleotti/PointCloudExport.mp4

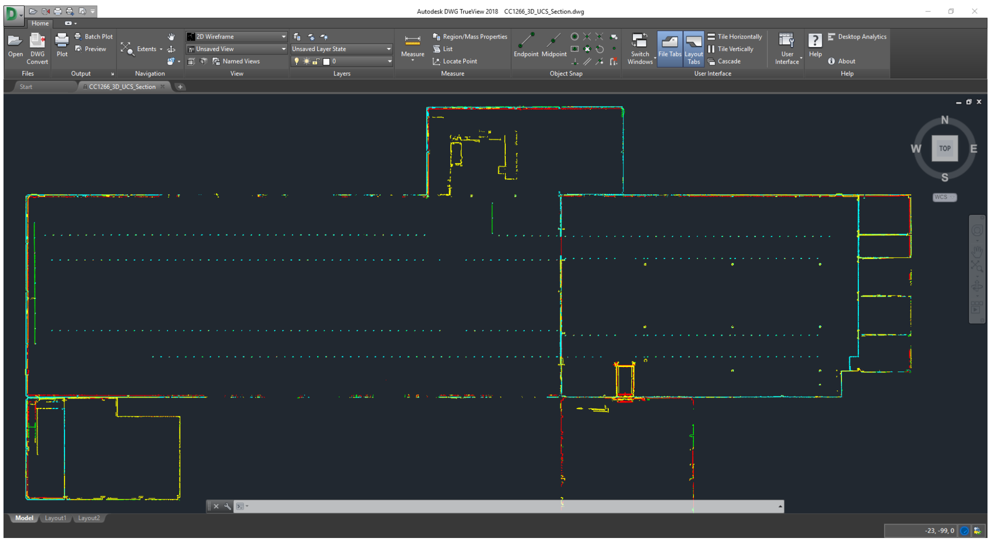


Figure 2.15: Example of generated floor plan at multiple heights.

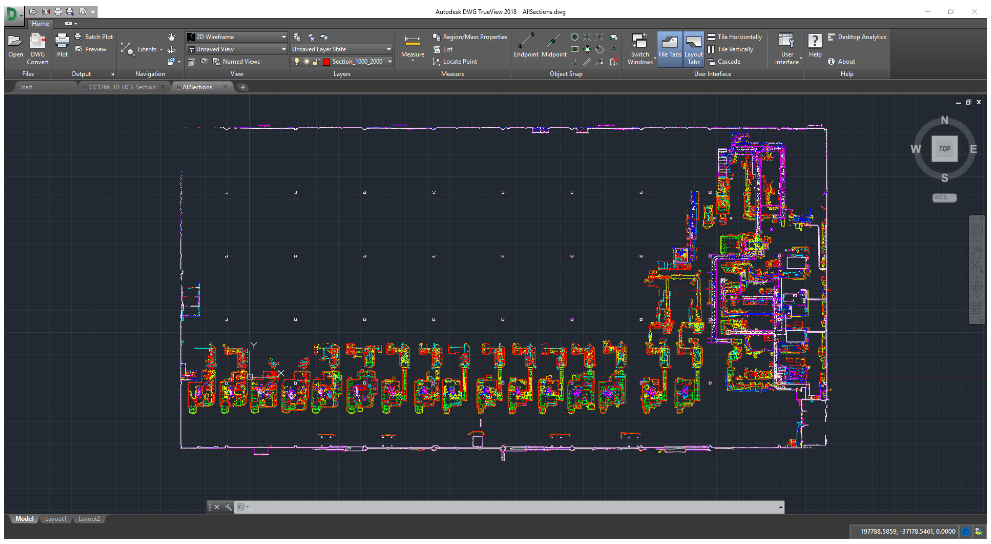


Figure 2.16: Example of generated floor plan at multiple heights.

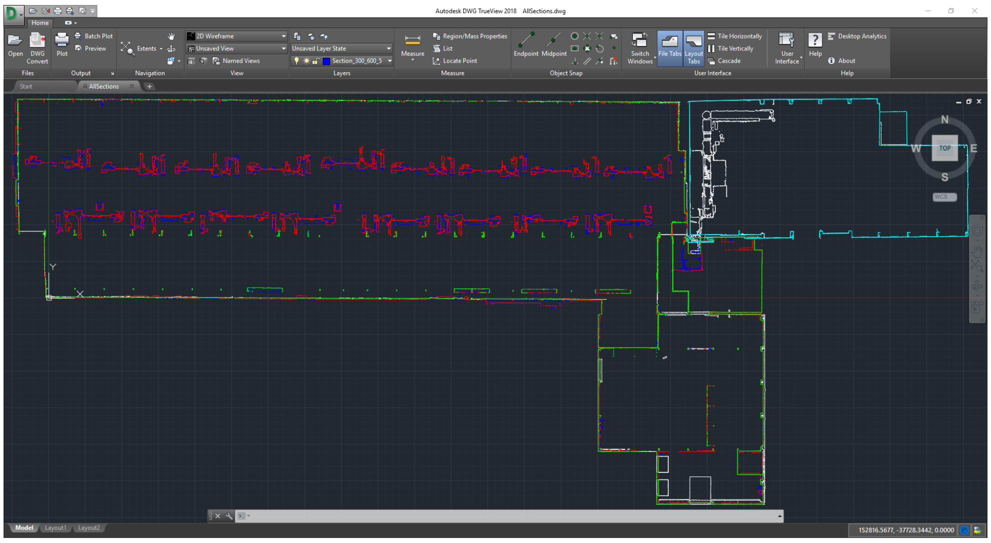


Figure 2.17: Example of generated floor plan at multiple heights.

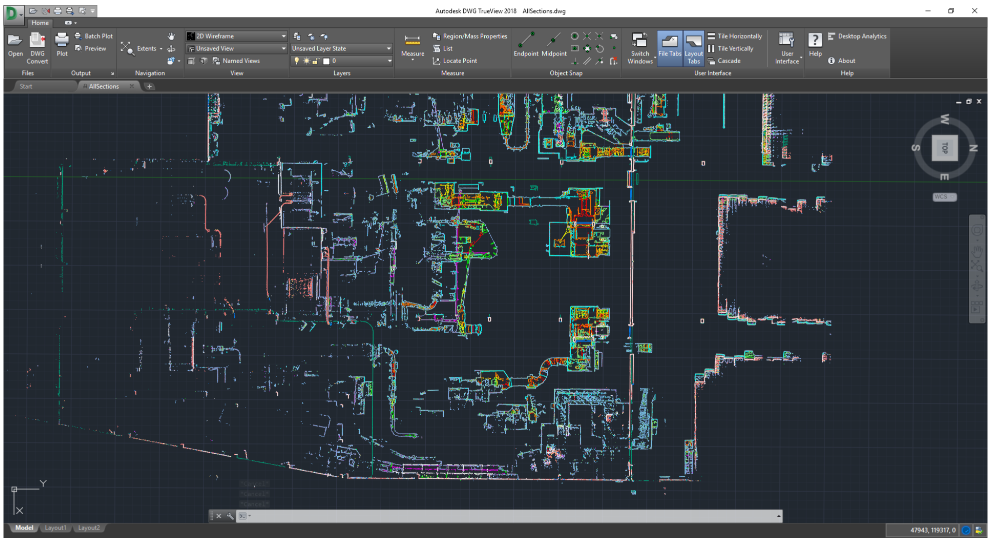


Figure 2.18: Example of generated floor plan at multiple heights.

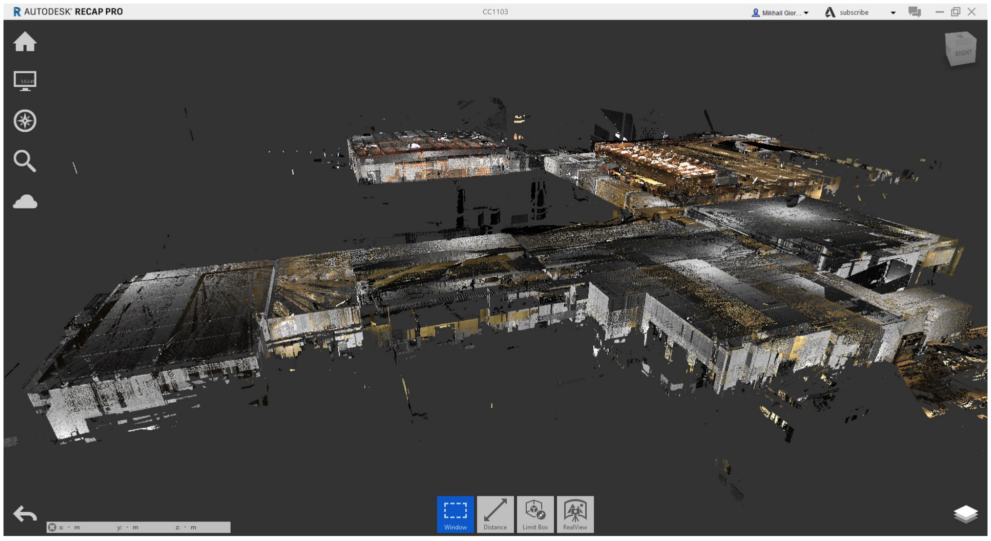


Figure 2.19: Example of exported point cloud.

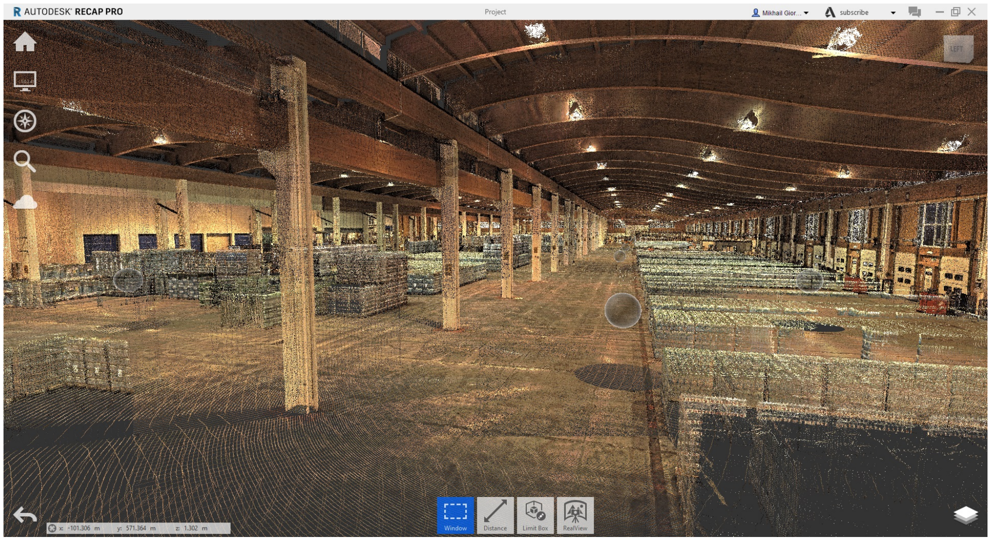


Figure 2.20: Example of exported point cloud.

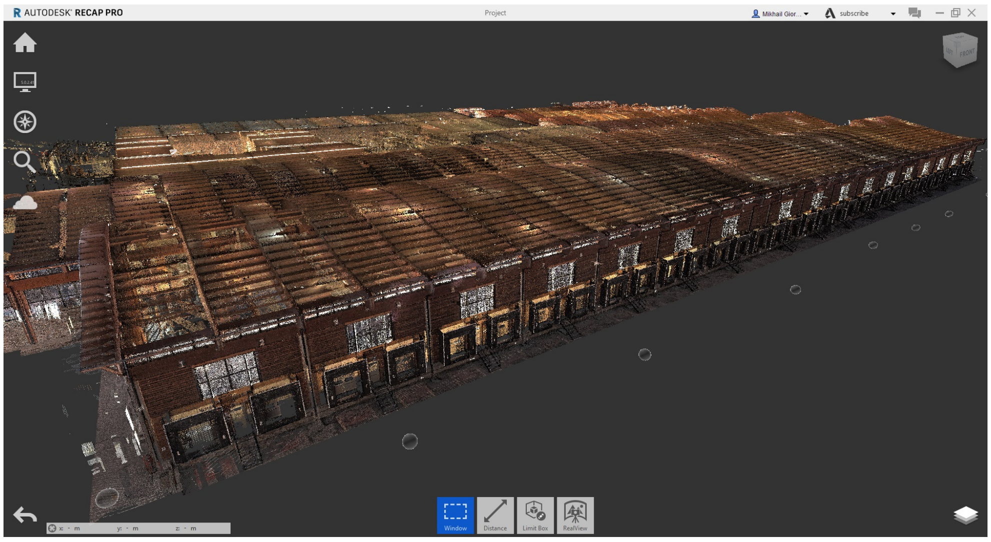


Figure 2.21: Example of exported point cloud.

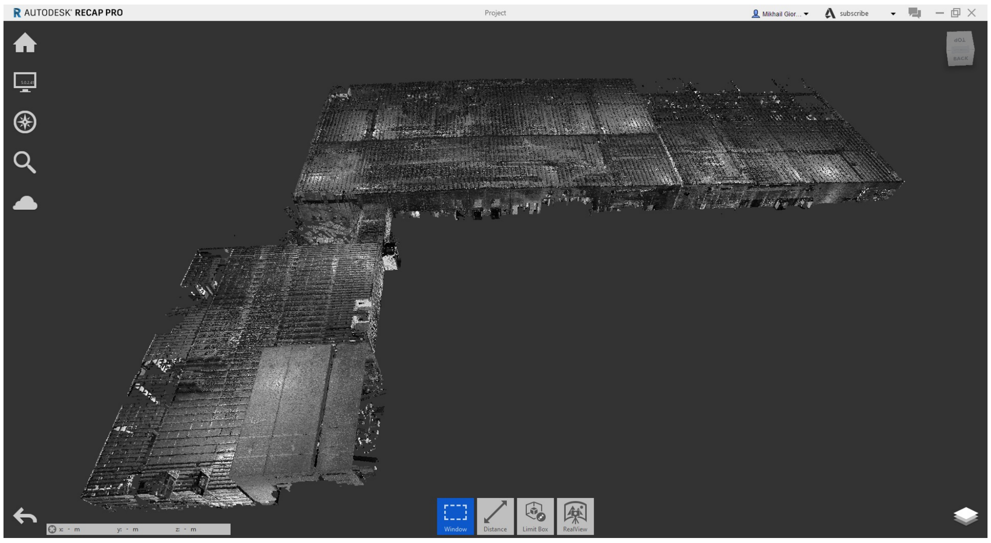


Figure 2.22: Example of exported point cloud.

task and it required to stop large warehouse and production areas, thus resulting in increased cost for the customer. Therefore, the extraction of AGV operation points becomes crucial, because it allows the design of AGV paths before the installation phase. Thanks to the proposed workflow, the extraction of operation points in a middle-sized warehouse takes less than one day as it only requires the user to manually determine the pallet positions based on rack sections. Figure 2.15 shows an example of a rack layout ready for manual extraction of operation points. Figures 2.23 and 2.24 show an example of manual extraction of operation points. However, manual extraction has the drawback of being error prone. Automatic techniques are being evaluated and will be part of future work.

## 2.6 Virtual plant deployment

By combining segmented ground, floor plan, cleaned point cloud and operation points, robot workcells position and AGV paths are accurately defined and validated by the technical office. In particular, AGV paths definition and robot workcells placing are performed in 2D exploiting floor plans at different heights, to support the traditional 2D CAD workflow of the company. Conversely, validation of robot workcells positions and AGV paths is performed in the 3D environment, exploiting the information provided by the 3D survey. Figure 2.25 shows the placing of two conveyors.

Chapter 5 presents a novel VR system that enables visualization of large-scale industrial datasets with real-time collision checking (at 60 fps) between the AGV model and the static point cloud model of the environment [24]. The proposed approach takes into account the local slope of the ground and supports the visualization of AGV safety zones. The VR system is then integrated with the company CAD software that is in charge of the design of AGV paths (Fig. 2.26). For a wider use, the collision checking system can also be used in desktop mode, without support to Virtual Reality (Figs. 2.27 and 2.28). A video of the integration with the existing CAD software is also available.[4]

[4]`www.ce.unipr.it/~aleotti/VirtualDeployment.mp4`

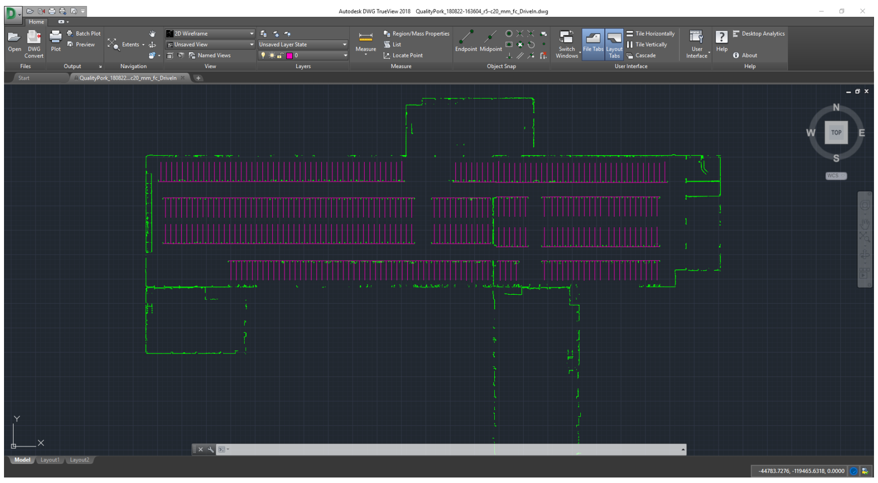


Figure 2.23: Example of manual extraction of operation points. Green segments show the generated floor plan, while purple segments are manually marked by the user as they identify operation points.

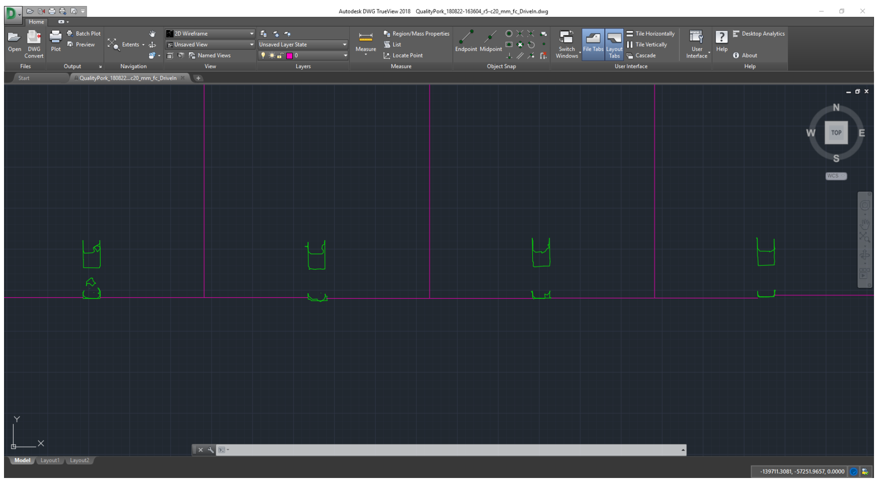


Figure 2.24: Example of procedure for manual extraction of operation points. Green objects show the position of rack columns, extracted exploiting the floor plan generation algorithm. The user determines operation points by drawing the bisector of the segment connecting two consecutive rack columns.

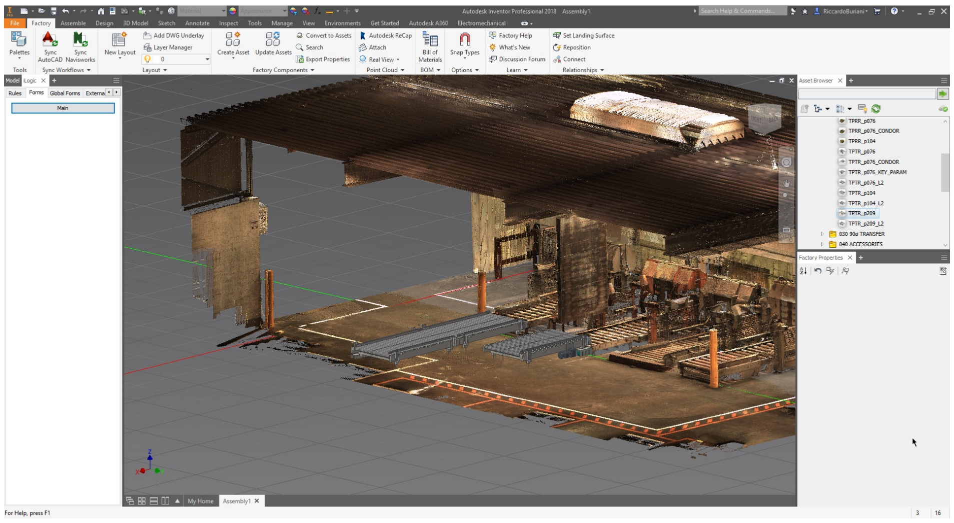


Figure 2.25: Example of conveyor placing. Geometric interferences between the conveyor model and the point cloud are visually inspected.

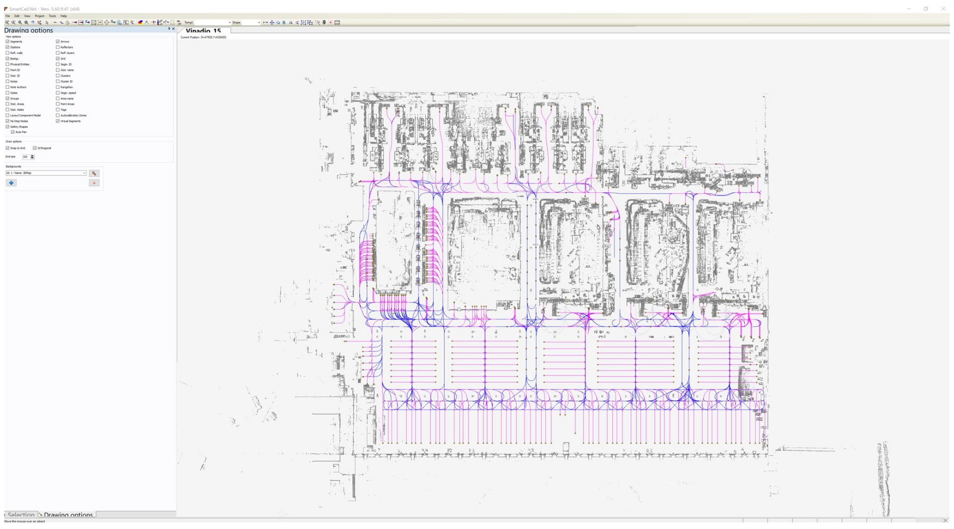


Figure 2.26: CAD software for AGV path definition. Generated floor plan (Section 2.5.2) is exploited as background for accurate path definition.

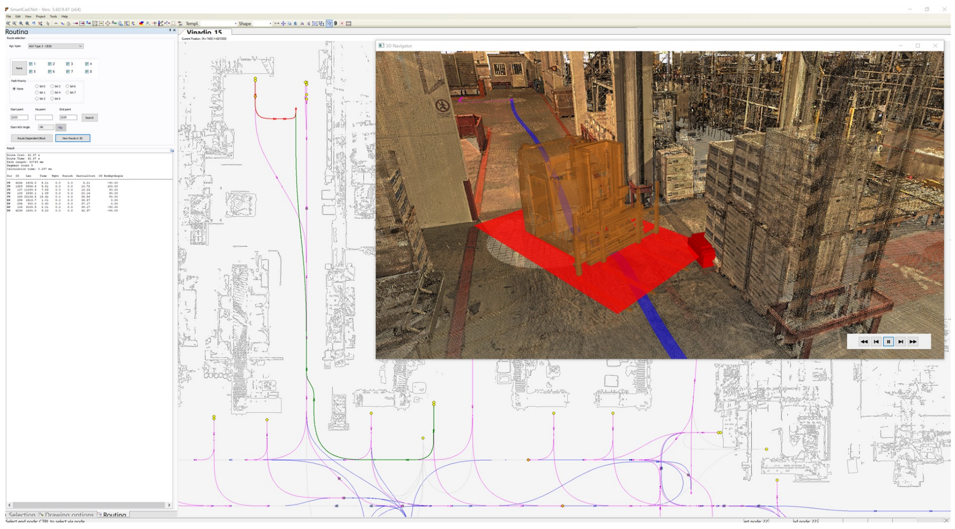


Figure 2.27: CAD software for AGV path definition with support to vehicle collisions. In case of collision, paths can be redrawn before the deployment.

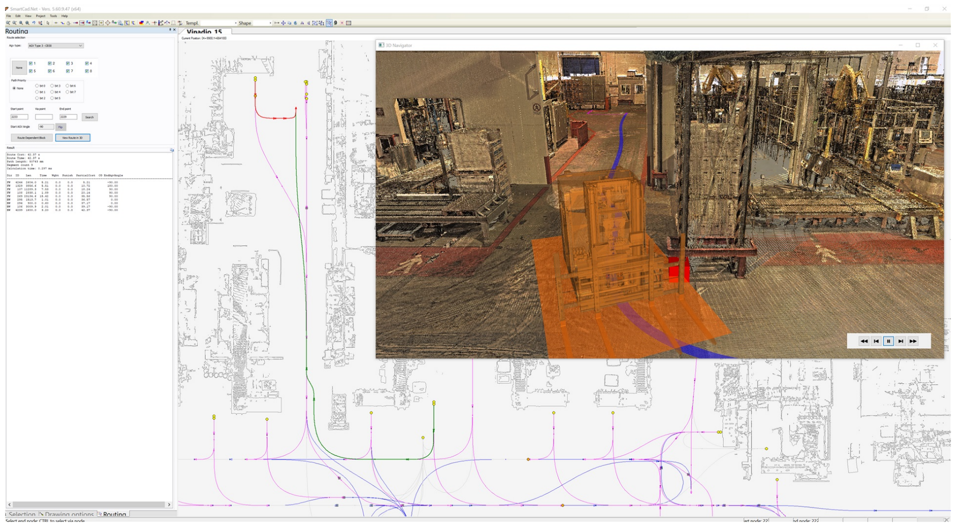


Figure 2.28: CAD software for AGV path definition with support to safety zone collisions. In case of collision, safety zones can be resized before the deployment.

# Chapter 3

# Ground Segmentation

## 3.1 Introduction

Extracting an accurate ground model from a 3D point cloud of a warehouse is essential to define and plan trajectories of Autonomous Guided Vehicles like driverless forklifts.

Complex industrial environments may contain terrains with non-planar regions and, therefore, techniques for planar surface extraction like RANSAC [5] are not appropriate as well as any method that assumes the presence of a dominant plane. This chapter presents an approach for ground segmentation in a registered large-scale point cloud with clutter acquired from a TLS survey (Section 2.2). The proposed method is capable of extracting the ground even if the input point cloud contains large and dense planar regions that do not belong to the ground. The algorithm first computes a robust estimate of a point belonging to the ground below each scan station. These points are then encoded into a fully connected graph and node conflicts, due to excessive slope, are detected and filtered out. Afterwards, a volume of interest that contains the ground is generated. Moreover, a fine-grained outlier removal process is applied by checking inconsistent normal vectors. Finally, a region growing phase, followed by gap filling, is performed to extract the ground. The chapter is organized as follows. Section 3.2 reviews the state of the art in ground segmentation and outlines the contribution of this work. Section 3.3 describes the proposed method for ground segmentation. Section 3.4

presents the experimental results in real large-scale industrial environments. Section 3.5 concludes the chapter.

## 3.2 Related works

Following the proposed categorization in [78] ground segmentation methods can be classified in four main categories that are briefly reviewed in this section.

Elevation maps approaches [73, 44, 52, 57] provide an efficient solution based on the computation of height histograms. However, they suffer from under-segmentation and require appropriate thresholds for sloped grounds that cannot always be easily determined.

Ground modeling methods [29, 14, 16, 15, 19, 31] refer to approaches that use a single sensor aboard vehicles and that provide a local ground estimation.

Methods based on relationship between adjacent points [76, 6, 47, 8, 45, 77] exploit sensor specific features like neighborhood relations between points and data ordering. Euclidean clustering and region growing algorithms are usually adopted for segmentation. The main drawback of these methods is that results highly depend on the choice of the point cloud seeds of the segments. In [77] a Progressive Morphological Filter was proposed to separate ground and nonground LIDAR measurements by gradually increasing the window size and by using elevation difference thresholds.

Markov Random Field (MRF) methods [78, 68] use a probabilistic terrain model. In particular, the work by Zhang et al. [78] shows a significant improvement over previously cited approaches by applying a multi-label MRF in polar coordinates and a loopy belief propagation method using a single LIDAR sensor.

Other previous works [1, 50, 54] have considered the extended problem of indoor reconstruction, which includes ground estimation as well as floor plan extraction. However, in these works ground estimation is usually performed in a simplified way by assuming the presence of horizontal planes. The approach proposed in this work can be classified as belonging to the class of methods based on relationship between adjacent points with the difference that it provides a robust estimation of the point cloud seeds. Indeed, the algorithm works on 3D data that are acquired by a TLS from multiple scan stations and it exploits this property to generate a robust estimation of ground points under each scan station. Moreover, the proposed approach scales linearly with the number of scans and it was evaluated in considerably larger environments than in previous works.

## 3.3 Ground segmentation

The algorithm consists of five sequential phases, that will be presented in the next subsections, and it requires a set of $n$ registered scans acquired from a TLS as input. The reference frame is located under the first scan station with z-axis opposite to the gravity vector. A TLS survey may contain different scans acquired from the same scan station. Taking multiple scans from the same scan station is quite common in highly dynamic environments where objects, like vehicles, are moving during the scan.

### 3.3.1 Extraction and filtering of virtual points on the ground

In the first phase one point belonging to the ground is computed from each scan by exploiting the peculiar characteristics of TLSs. Indeed, it is reasonable to assume that any region of space below a scan station is almost planar and that it contains the ground, since the sensor is fixed on a tripod that is placed in an empty area. However, as the terrestrial laser scanner has a limited vertical field of view $\Phi$ there is a blind spot below the sensor (Fig. 3.1). Hence, no point can be selected in the blind spot. Therefore, a point on the ground is computed in each scan by applying a local floor estimation procedure using the measured points in the neighborhood of the scan station. Points computed by this procedure are called "virtual points", since they do not belong to the measured point cloud.

In particular, a fast histogram-based method is adopted to determine the height of the floor below the sensor by considering all the measured points from the laser beams in the interval $\Delta\phi = [\phi_1, \phi_2]$ (Fig. 3.1). Each bin of the height histogram (Fig. 3.2) counts the number of points in the same height range, where $\Delta_h = 5$ mm is the quantization step. For example, if a TLS with a vertical field of view of 290° is placed on a tripod at 1.7 m height with $\Delta\phi = 10°$, about 750k points will be considered, at

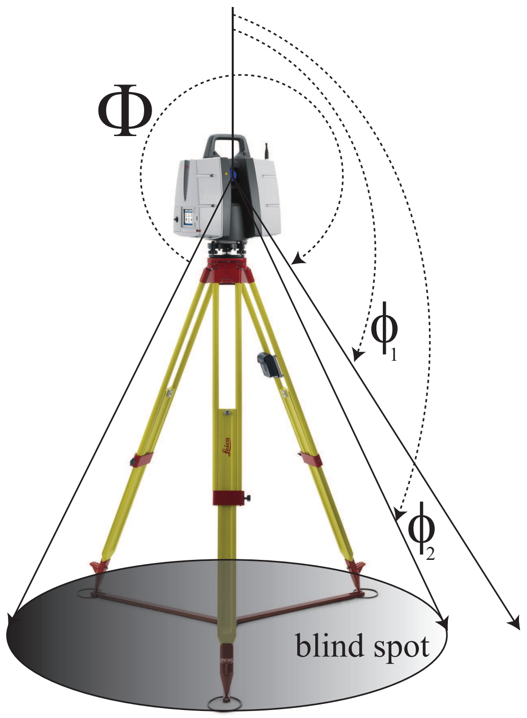


Figure 3.1: Local plane estimation is performed using the point cloud in the neighborhood of the scan station within the angular interval $\Delta\phi = [\phi_1, \phi_2]$ as the TLS has a blind spot.

a standard resolution of 1.2 mrad, which is ample to achieve an accurate local floor estimation. Most of these points are likely to belong to the floor.

The height histogram is processed to find the maximum peak. The peak is a subset of adjacent bins defined as follows. Let $\{b_{0_j}, \dots, b_{m_j}\}$ be the value of the bins of the histogram of scan $j$, and $\{h_{0_j}, \dots, h_{m_j}\}$ the associated heights. The algorithm finds the bin $p_j$ with the maximum value and performs a recursive $k$-nearest neighbor search to find all the bins $i_j$ with $b_{i_j} > \beta b_{p_j}$, where $\beta \in [0, 1]$. The bandwidth of the peak of scan $j$ is defined by the set of contiguous bins within the leftmost index $u_j$ and rightmost index $v_j$ found in the recursive search. Let the peak of the histogram being

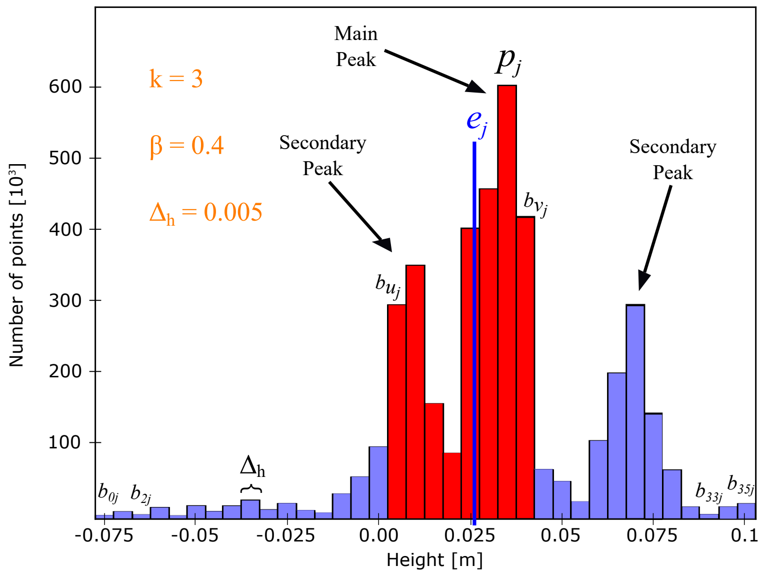


Figure 3.2: Example of height histogram. The peak of the histogram is composed by red bins. Secondary peak on the left is included in the main peak, while secondary peak on the right is not part of the main peak due to the recursive $k$-nearest neighbor search.

composed by bins with values $\{b_{u_j}, \ldots, b_{v_j}\}$, associated to heights $\{h_{u_j}, \ldots, h_{v_j}\}$, where $w_j = v_j - u_j + 1$ is the peak width. The height of the virtual point in scan $j$ (coordinate $z$) is computed as a weighted average, defined as

$$e_j = \frac{\sum_{i=u_j}^{v_j} b_i h_i}{\sum_{i=u_j}^{v_j} b_i} \tag{3.1}$$

The virtual point $g_j$ in scan $j$ is then defined as having the same planar coordinates of the TLS center $[x_j, y_j, z_j]^T$, i.e.

$$g_j = [g_{j_x}, g_{j_y}, g_{j_z}]^T \triangleq [x_j, y_j, e_j]^T \tag{3.2}$$

The computation of all virtual points $\{g_0, \ldots, g_{n-1}\}$ of the scans is performed in parallel. An accuracy score $\alpha_j \in [0, 1]$ for each virtual point is also computed.

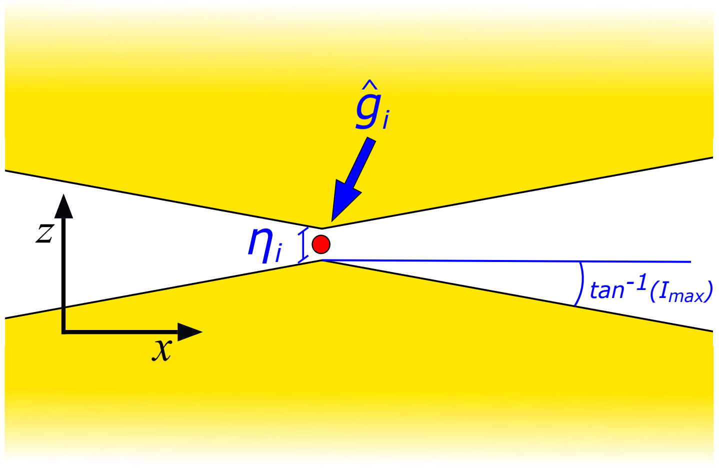


Figure 3.3: Volume $V_i$ (white region). Vertical section view.

Let $w^* = \min_{j\in 0\ldots n-1}\left(w_j\right)$ be the minimum peak width across all the $n$ scans.
Let $b^* = \max_{j\in 0\ldots n-1}\left(b_{p_j}\right)$ also be the maximum peak height across all the $n$ scans.
Accuracy $\alpha_j$ of the virtual point $g_j$ is then defined as

$$\alpha_j = \frac{1}{2}\frac{b_{p_j}}{b^*} + \frac{1}{2}\frac{w^*}{w_j} \tag{3.3}$$

Outlier detection and filtering of the extracted virtual points $\{g_0, \ldots, g_{n-1}\}$ on the ground are then performed to improve robustness and to reduce as much as possible the number of false positives, even at the expense of removing some inlier virtual points. Indeed, losing some inlier points is less critical than keeping wrong seeds. First, if more than one virtual point has been generated for the same scan station, due to multiple scans taken from it, the algorithm selects only the virtual point with the highest accuracy (scans are not removed). Then, each remaining virtual point $g_i$ is treated as a node of a complete (fully connected) graph $G = (V, E)$, with vertex-set $V$, edge-set $E$ and $|G| \leq n$. Each edge $(i, j)$ connecting nodes $g_i = [x_i, y_i, e_i]^T$ and $g_j = [x_j, y_j, e_j]^T$ is marked as "inconsistent" if the slope of the segment connecting the two nodes is greater than a threshold $I_{max}$, i.e. $I_{i,j} > I_{max}$. The slope of a segment

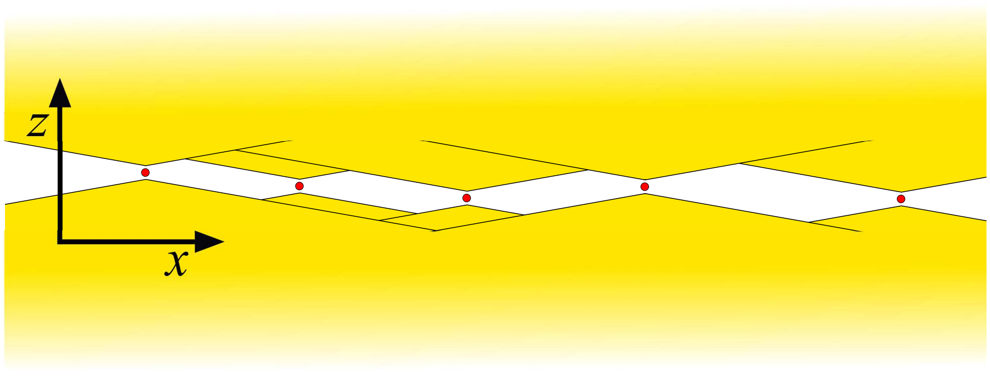


Figure 3.4: Volume $V$ (white region). Vertical section view. Points of the point cloud in $V$ provide a coarse point cloud segmentation of the ground. Points of the point cloud inside the yellow volume are considered outliers. Red dots indicate the virtual points on the ground.

is computed as

$$I_{i,j} = \frac{|e_i - e_j|}{\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}} \tag{3.4}$$

A "degree of inconsistency" $|\Gamma(g_i)|$ is assigned to each node, computed as the number of inconsistent edges connected to that node. The algorithm finds the node $\bar{g}$ with the highest degree of inconsistency among those with $|\Gamma(g_i)| > 1$, i.e

$$\bar{g} = \underset{g_i \text{s.t.} |\Gamma(g_i)|>1}{\arg\max} |\Gamma(g_i)| \tag{3.5}$$

and removes $\bar{g}$ from the graph, decreasing also the degree of inconsistency of all its neighboring nodes. Node removal, which is aimed at removing nodes that have a higher probability of being outliers, is repeated until no more nodes with $|\Gamma(g_i)| > 1$ are found. Finally, the graph is further reduced to remove residual inconsistencies by deleting all remaining nodes $g_i$ with $|\Gamma(g_i)| = 1$ in a single iteration. The output of this phase is a reduced graph, whose nodes represent the estimated inlier virtual points, named $\{\hat{g}_0, \ldots, \hat{g}_l\}$.

### 3.3.2 Coarse segmentation of ground points

Starting from the robust estimation of the set of virtual points located below each scan station, a coarse ground segmentation is performed by keeping points of the point cloud contained in a volume $V$ defined as the intersection of a set of sub-volumes, i.e. $V = \bigcap_i V_i$. Each sub-volume $V_i$, illustrated in Fig. 3.3, is generated from a virtual point $\hat{g}_i$ and it contains all points $[x, y, z]^T$ of the point cloud lying within a double cone, i.e. satisfying the following condition

$$\hat{g}_{i_z} - f_i(x, y) \leq z \leq \hat{g}_{i_z} + f_i(x, y) \tag{3.6}$$

with $f$ defined as the nappe

$$f_i(x, y) = \frac{\eta_i}{2} + I_{max}\sqrt{x^2 + y^2} \tag{3.7}$$

where $I_{max}$ accounts for the uncertainty of the ground slope, and $\eta_i = w_i \Delta_h$ is an offset that accounts for the uncertainty associated to a virtual point $\hat{g}_i$ where, as defined in section 3.3.1, $w_i$ is the width of the peak, while $\Delta_h$ is the quantization step of the histogram. Volume $V$ is shown in Fig. 3.4. Examples of coarse ground segmentation are shown in Figs. 3.5 and 3.6. At this phase of the algorithm points are still organized into separate scans. To speed up nearest neighbor search in the following phases, all points in the scans are hereafter organized into a single regular grid of cells on the $xy$ plane. The grid-based representation, not only enables a fast proximity search, but it also enables parallel processing of the whole point cloud. Also, in this phase the point cloud is down-sampled to reduce the computational time of the following phases, depending on the precision requirements. In this work a 5 mm spatial resolution has been used.

### 3.3.3 Normal filtering

A more fine-grained outlier removal procedure is then applied by checking inconsistent normal vectors. Normal vectors are estimated for each point of the coarse ground segmentation. Point cloud normals are computed from local plane fitting with 5 cm radius. Points whose normal vector forms an angle with the $z$-axis that exceeds a

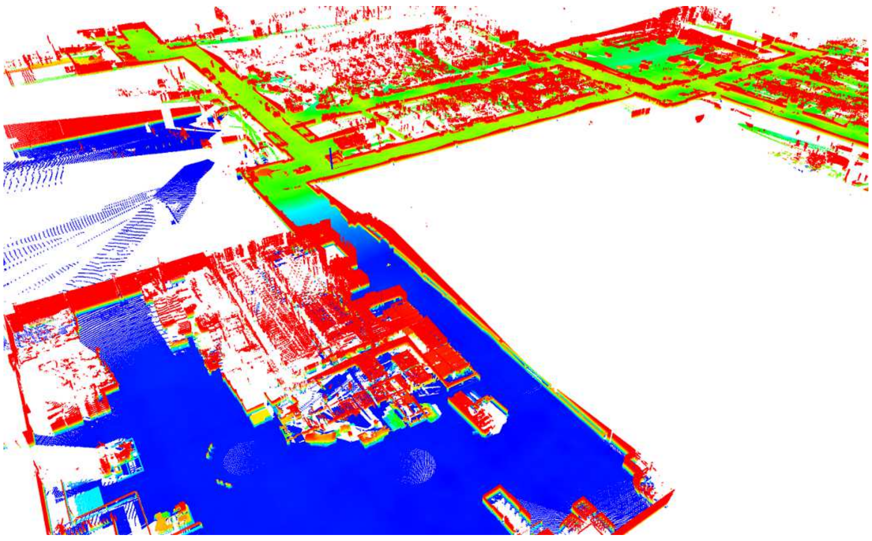


Figure 3.5: Example of coarse ground segmentation. Parameter $I_{max}$ has been increased to $0.1$ (10% slope) to better appreciate the effect on volume $V$.

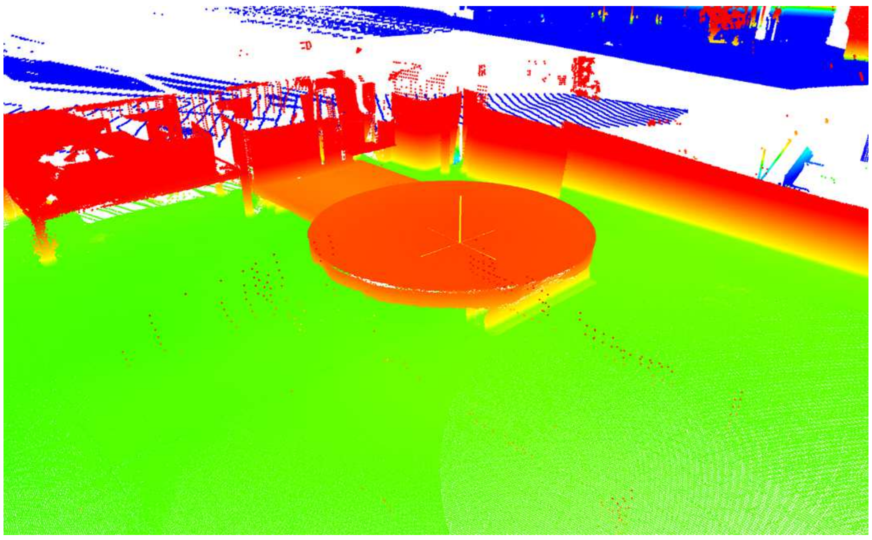


Figure 3.6: Example of coarse ground segmentation.

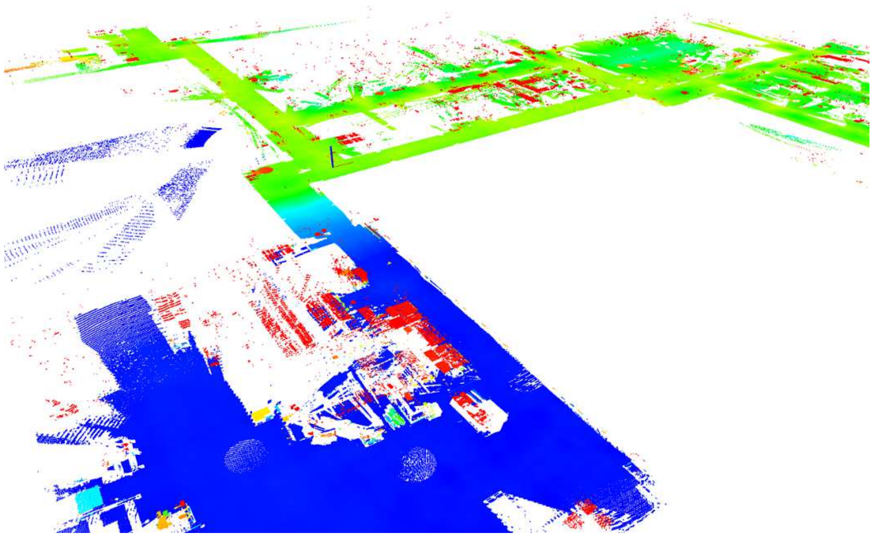


Figure 3.7: Example of normal filtering (sec. 3.3.3). All points belonging to non-planar surfaces have been removed.

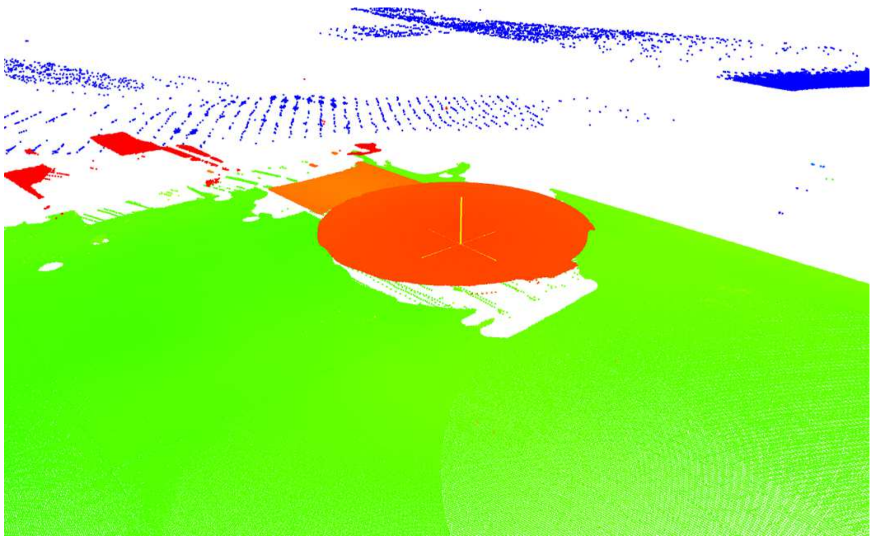


Figure 3.8: Example of normal filtering (sec. 3.3.3). Planar surfaces not belonging to the ground are still present, however they got separated from the ground.

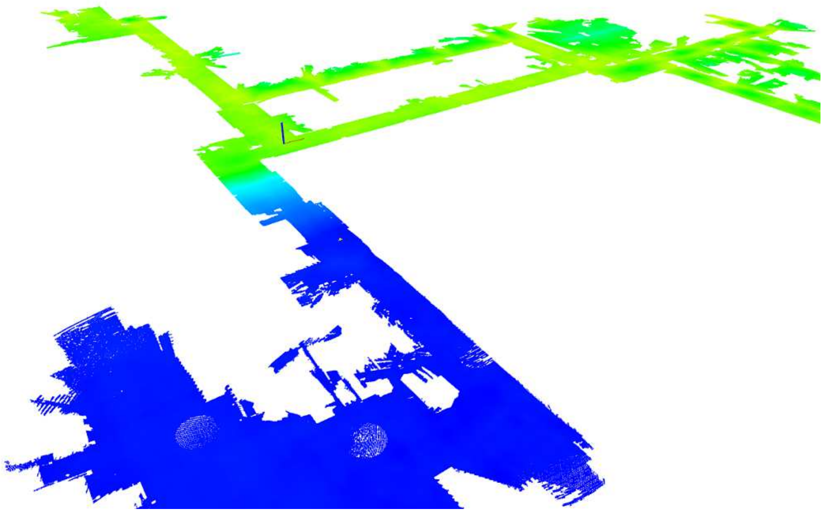


Figure 3.9: Example of region grow (sec. 3.3.4). All planar surfaces not belonging to the floor have been removed, as well as non-connected regions.

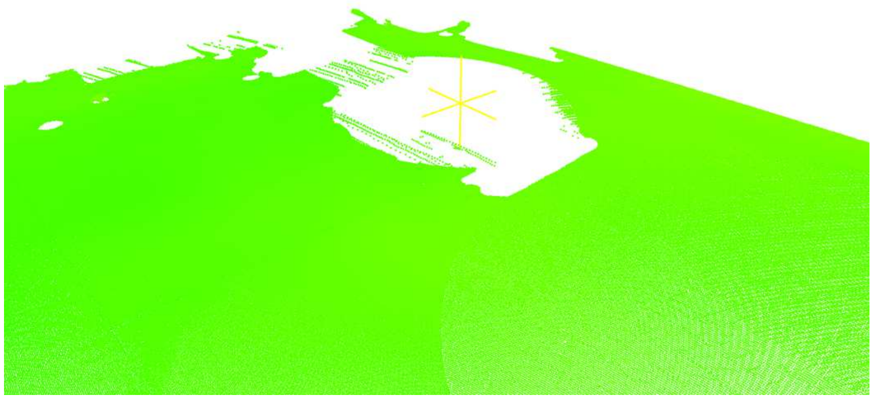


Figure 3.10: Example of region grow (sec. 3.3.4). The planar surface displayed in red in Fig. 3.8 has been removed.

threshold $\theta_{max}$ are discarded. Threshold $\theta_{max}$ is set higher than $\tan^{-1}(I_{max})$ due to measurement noise, by multiplying $I_{max}$ by a factor $\nu > 1$, i.e.

$$\theta_{max} = \tan^{-1}(\nu I_{max}) \tag{3.8}$$

Appropriate threshold values for the TLS sensor adopted in this work have been determined empirically as $I_{max} = 0.02$ and $\nu = 5$. Examples of the normal filtering phase are illustrated in Figs. 3.7 and 3.8. It can be noticed that all points belonging to non-planar surfaces have been filtered out, and that planar regions not belonging to the ground have been separated. Hence, such planar regions will not pass the region growing phase described in section 3.3.4.

### 3.3.4 Association of virtual points to measured points and region grow

In this phase each virtual point $\{\hat{g}_0, \ldots, \hat{g}_l\}$ is first associated to one of the neighbor measured points in $V$. Although virtual points belong to the TLS blind spots, there are plenty of measured points close to them in the entire point cloud, since the blind spots are observed from other scans. In particular, given the set of measured points $\{p_0, \ldots, p_N\}$ with $p_i \in V$ and $p_i = [p_{i_x}, p_{i_y}, p_{i_z}]^T$, each virtual point $\hat{g}_i$ is mapped to the nearest measured point $g_i^*$ on the $z$-axis (if any) within a horizontal distance $\Delta_{xy}$, which represents the radius of the cone with vertex at the TLS center, with half-angle $\pi - \phi_1$ and height $|\hat{g}_{i_z} - z_i|$, where $z_i$ is the vertical coordinate of the TLS center, i.e.

$$\begin{aligned} g_i^* &= \underset{p_i \text{ s.t. } d_{xy} < \Delta_{xy}}{\arg\min} |p_{i_z} - \hat{g}_{i_z}| \\ d_{xy} &= \sqrt{(\hat{g}_{i_x} - p_{i_x})^2 + (\hat{g}_{i_y} - p_{i_y})^2} \\ \Delta_{xy} &= |\hat{g}_{i_z} - z_i| \tan(\pi - \phi_1) \end{aligned} \tag{3.9}$$

The decision function in Eq. 3.9 performs better than a standard Euclidean distance in $R^3$ because the uncertainty level in the estimation of the planar coordinates of the virtual points, which stems from the point cloud registration phase, is larger than the uncertainty level of the vertical coordinate $e_i$ of the virtual points, which stems from

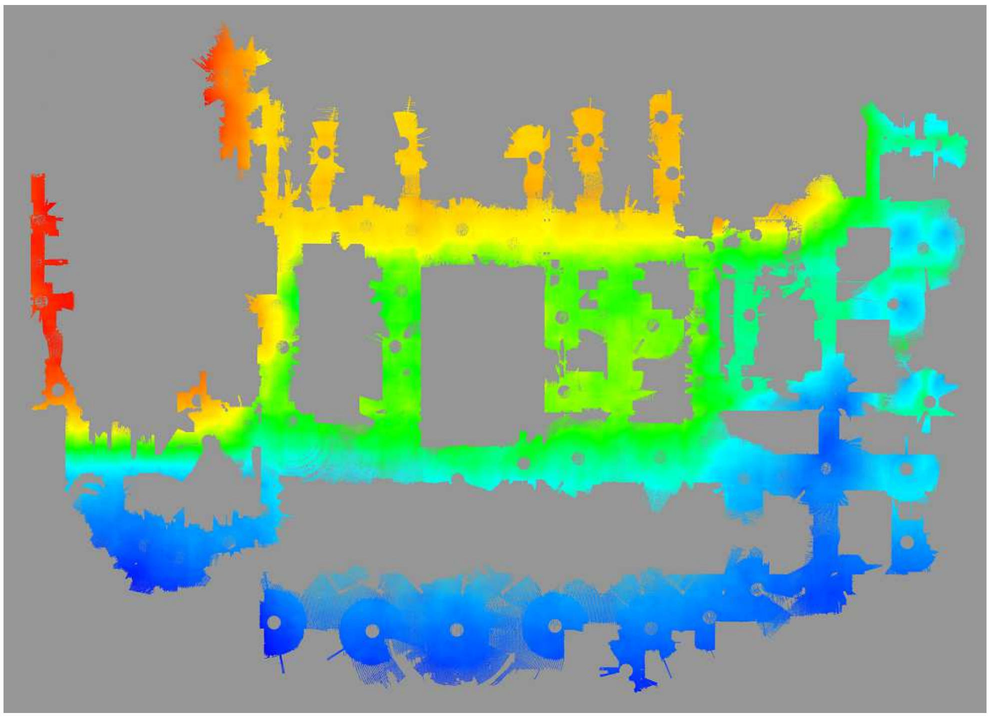


Figure 3.11: *Dataset*04 (top view). Rasterization after region grow.

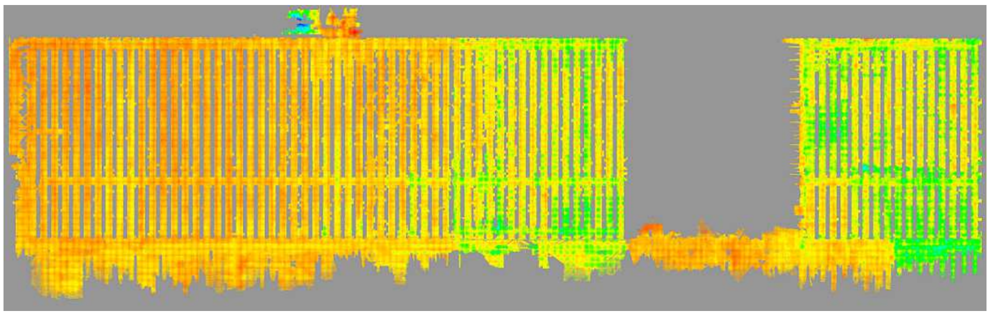


Figure 3.12: *Dataset*05 (top view). The largest dataset where the proposed approach has been evaluated, with more than 70.000 m$^2$ of surface and an initial amount of over 15 billions of points.

Table 3.1: Datasets used in the experimental evaluation

| **Dataset** | **Company** | **Location** | **Scans** | **Resolution [mrad]** | **Points** | **Size [m$^2$]** | **Proc. time [s]** |
|---|---|---|---|---|---|---|---|
| *Dataset*00 | Elettric 80 S.p.A. | Viano (Italy) | 8 | 1.2 | 87 494 103 | 735 | 17 |
| *RND* | Elettric 80 S.p.A. | Viano (Italy) | 11 | 1.2 | 106 417 059 | 735 | 22 |
| *Dataset*01 | *Food* | North Dakota | 47 | 1.2 | 446 273 160 | 4 200 | 114 |
| *University* | | Parma (Italy) | 48 | 0.6 | 1 441 032 861 | 17 675 | 169 |
| *Dataset*02 | *Tissue* | Nevada | 54 | 1.2 | 497 805 241 | 23 160 | 131 |
| *Dataset*03 | *Food* | Italy | 58 | 1.2 | 544 864 429 | 17 850 | 144 |
| *Dataset*04 | Acqua Sant'Anna | Italy | 117 | 1.2 | 1 031 886 921 | 53 610 | 285 |
| *Dataset*05.1 | Empresas Carozzi | Chile | 372 | 1.2 | 3 327 399 675 | 14 105 | 737 |
| *Dataset*05.2 | Empresas Carozzi | Chile | 1263 | 1.2 | 11 928 266 084 | 49 600 | 2783 |
| *Dataset*05 | Empresas Carozzi | Chile | 1641 | 1.2 | 15 404 201 936 | 77 035 | 3583 |

the procedure explained in section 3.3.1. The set of mapped points $\{g_0^*, \ldots, g_q^*\}$, with $q \leq l$, are likely to belong to the ground and are used as seeds for a conditional region growing procedure, constrained by point distance and segment slope. Indeed, a point $p_i$ is recursively added to region $R$, that is the final ground model, if $\exists p_k \in R$ which satisfies the two conditions

$$\begin{aligned} &||p_k - p_i||_2 < \tau && \wedge \\ &\frac{|p_{k_z} - p_{i_z}|}{\sqrt{(p_{k_x} - p_{i_x})^2 + (p_{k_y} - p_{i_y})^2}} < I_{max} \end{aligned} \tag{3.10}$$

where $\tau = 5$ cm. Results of the region growing phase are shown in Figs. 3.9 and 3.10. The region growing phase has been optimized to handle the large scale point cloud by exploiting a parallel implementation with different seeds per thread and a caching policy. In particular, a binary vector $\xi$ of length $N$, shared among threads, is stored in RAM and updated so that

$$\xi_i = \begin{cases} 1 & \text{if } p_i \in R \\ 0 & \text{otherwise} \end{cases} \tag{3.11}$$

whereas the full point cloud is stored in the hard disk and chunks of points are loaded and unloaded as requested.

### 3.3.5 Gap filling

After region growing, the set of points $R$ representing the ground is converted to a height grid data structure with resolution $\Delta_r$. If multiple points fall inside the same

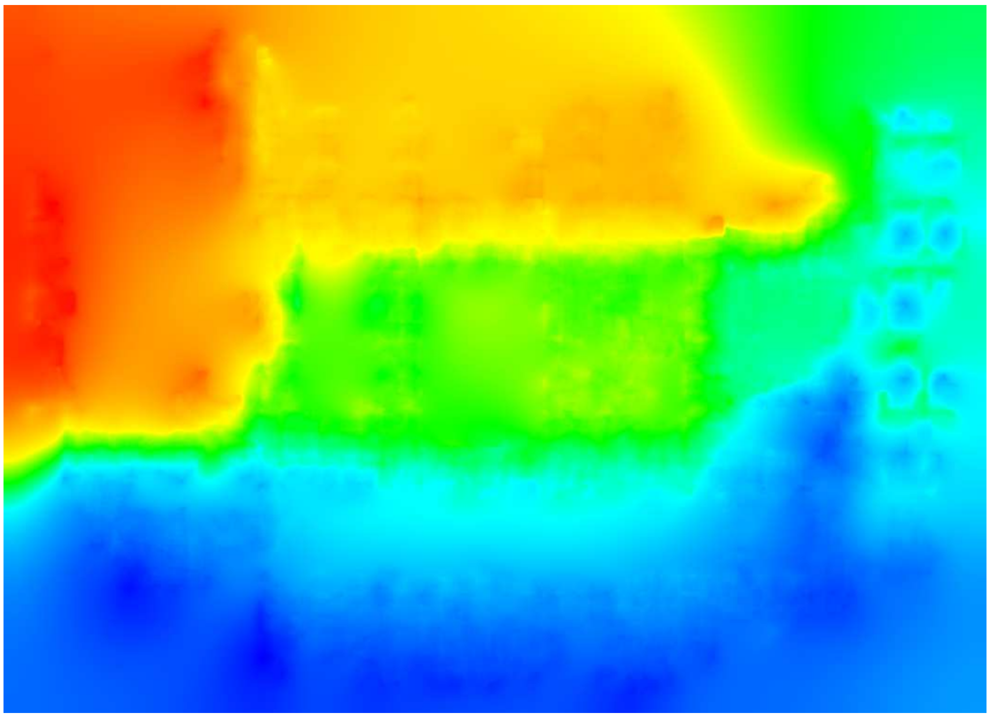


Figure 3.13: *Dataset*04 (top view). Results after SAGA filling [18].

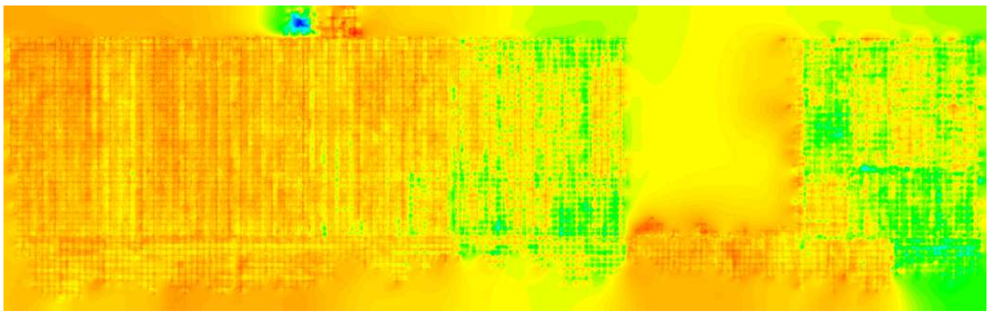


Figure 3.14: *Dataset*05 (top view). Results after SAGA filling [18].

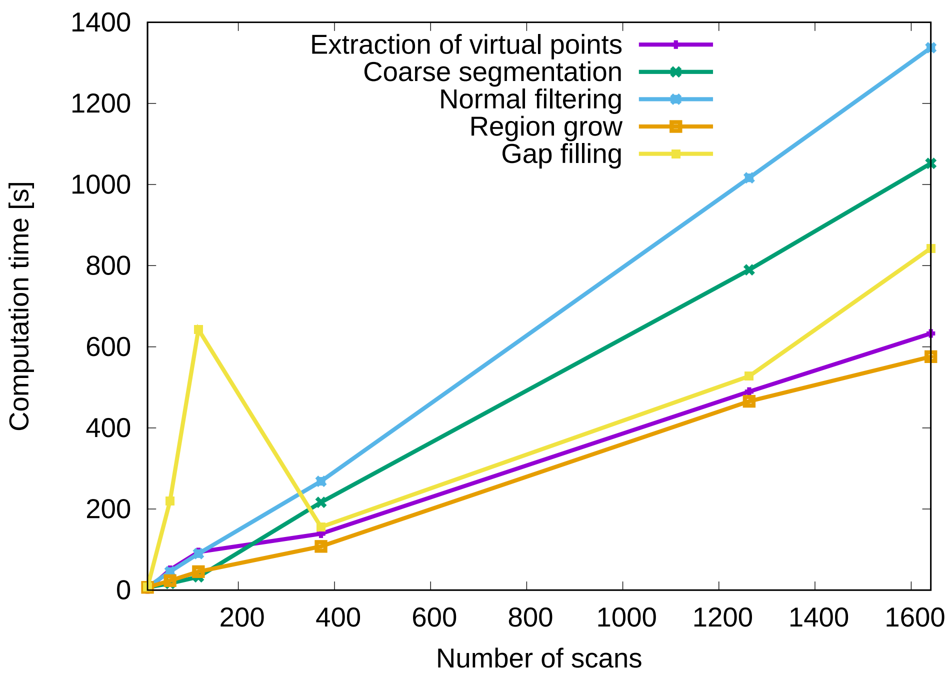


Figure 3.15: Computation time of each phase of the algorithm with respect to the number of scans. “Gap filling” phase does not have a linear complexity.

cell of the grid the height of the cell is set as the height of the lowest point, thus reducing the influence of points in $V$ due to moving objects on the ground during the scan, which may have slightly higher altitude. Examples of height grids are shown in Figs. 3.11 and 3.12. Gaps in the grid are then filled by using SAGA-GIS software library [18]. Examples of the resulting dense grid are shown in Figs. 3.13 and 3.14. The final result is the segmented ground, which can be used in industrial applications requiring an accurate model.

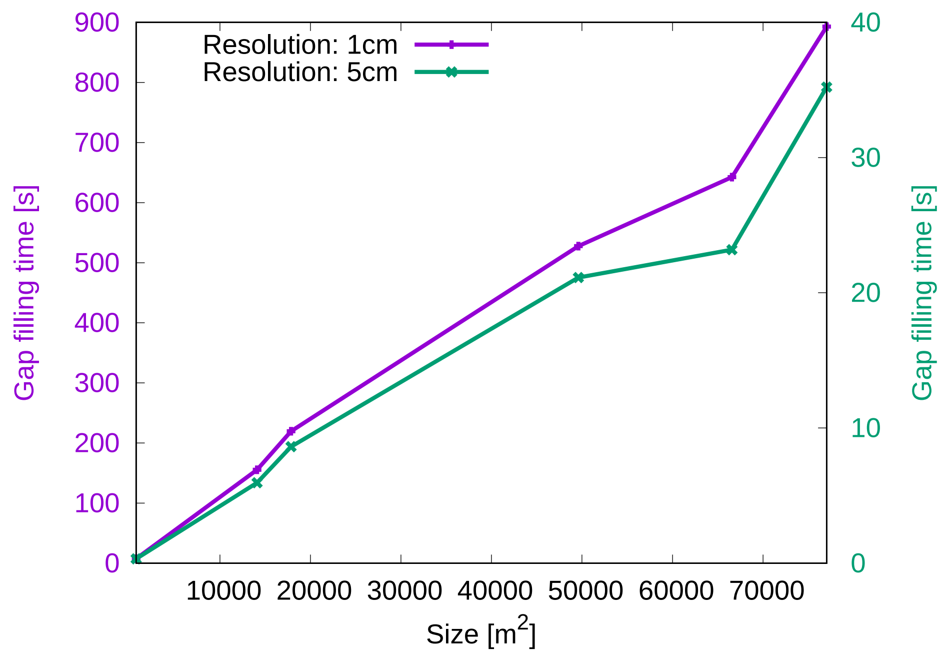


Figure 3.16: Computation time of the gap filling phase with respect to the size of the environment with $\Delta_r = 1$ cm (left y axis) and $\Delta_r = 5$ cm (right y axis).

## 3.4 Experimental results

Data processing has been performed on a desktop Intel i7-5960x (3.00GHz) CPU with 64GB RAM, a Solid State Drive and a Nvidia Titan X 12GB graphics card. Experiments have been performed in several environments mainly from food, beverage and tissue industrial warehouses (which contain racks for pallet storage or production lines) with different sizes across Europe, South America and NAFTA. In particular, Empresas Carozzi S.A. and Acqua Sant'Anna are leading companies in industrial warehousing for food and beverage respectively. Table 3.1 summarizes information about the datasets and some experimental results. Environments size ranged from a single industrial building of $10^3$ m$^2$ (*RND*) to a large food warehouse of over 70 000

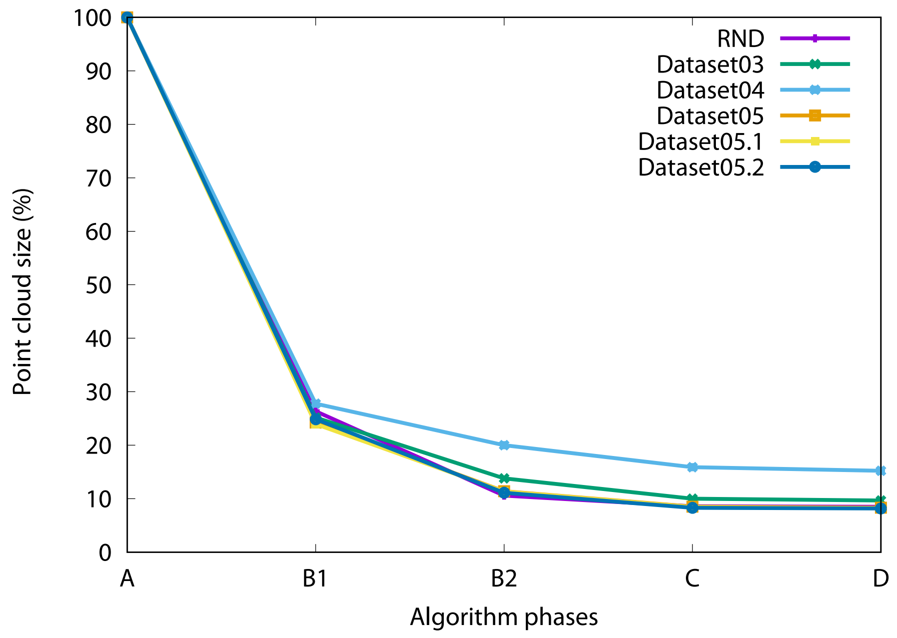


Figure 3.17: Point cloud size after each phase. The coarse segmentation phase (B) has the highest impact and it has been split into sub-phase B1 (actual coarse segmentation) and sub-phase B2 (down-sampling).

$m^2$ (*Dataset*05). The number of scans ranged from 11 to 1641, containing a total number of points from about $10^8$ to over $10^{10}$. Most of the TLS surveys have been performed at 1.2 mrad resolution. The largest scenario (*Dataset*05) has been processed both in its entirety and in two separate parts (*Dataset*05.1 and *Dataset*05.2) to assess the performance of the ground segmentation algorithm on various input sizes. Moreover, one of the datasets (*University*) acquired at the Department of Engineering and Architecture at the University of Parma, contains both indoor and outdoor scans, thus showing the applicability of the proposed approach to heterogeneous environments. This dataset has been obtained with scan stations located at higher distances and with higher resolution (0.6 mrad). It is important to notice that the proposed algorithm was evaluated in considerably larger environments than previous approaches. Indeed, applicability of previous methods in large scale environments has still to be proved.

Table 3.2: Comparison of the proposed approach with the progressive morphological filter (PMF) [77] distributed within the Point Cloud Library (PCL).

| **Algorithm** | **Accuracy** | **Precision** | **Recall** | **F1 Score** | **Processed Points** | **Proc. time** |
|---|---|---|---|---|---|---|
| Proposed | 0.9997 | 0.9999 | 0.9989 | 0.9994 | 87 494 103 | 17s |
| PMF @10mm | - - - | - - - | - - - | - - - | 23 879 127 | ≥ 1 day |
| PMF @25mm | 0.9887 | 0.9960 | 0.9624 | 0.9789 | 4 576 871 | 2h 54m 52s |
| PMF @50mm | 0.9965 | 0.9910 | 0.9961 | 0.9936 | 1 274 238 | 12m 18s |
| PMF @100mm | 0.9947 | 0.9815 | 0.9995 | 0.9904 | 340 936 | 58s |
| PMF @200mm | 0.9882 | 0.9590 | 0.9995 | 0.9788 | 89 280 | 5s |

Quantitative evaluation has been performed in terms of computation time and percentage of points processed in each phase. The algorithm was able to process the largest dataset in less than one hour. Fig. 3.15 shows that the total computation time grows linearly with the number of scans in all phases, except for gap filling. The computation time of the gap filling phase is linear with respect to the plant size, as shown in Fig. 3.16. It can be observed that the slowest phase is normal filtering. Fig. 3.17 shows the point cloud size after each phase. The coarse segmentation phase, before down-sampling, has the highest impact on reducing the number of points, indeed, after coarse segmentation the total number of points decreases, on average, to about 30%. Down-sampling at 5 mm spatial resolution, as described in section 3.3.2, further reduces the total number of points to about 15%.

Figs. 3.18, 3.19, 3.20, 3.21, 3.22 show the segmented ground, before the gap filling phase. The segmented ground points are displayed in color scale, ranging from blue (lowest height) to red (highest height). Dataset *University* and *Dataset*04 in Fig. 3.20 are two examples of environments with almost constant slope. Dataset *RND* in Fig. 3.22 and *Dataset*01 in Fig. 3.18 exhibit a more irregular ground (i.e. with bumps). Having an accurate segmentation of a ground with irregular shape is very important for example in industrial applications involving Autonomous Guided Vehicles. Conversely, *Dataset*03 in Fig. 3.19 shows a nearly planar environment, except for a highly sloped corridor, while *Dataset*02 has been successfully extracted from a warehouse despite many moving objects on the ground (scraps and rolls of paper). Finally, the proposed approach was compared with the progressive morphological filter (PMF)

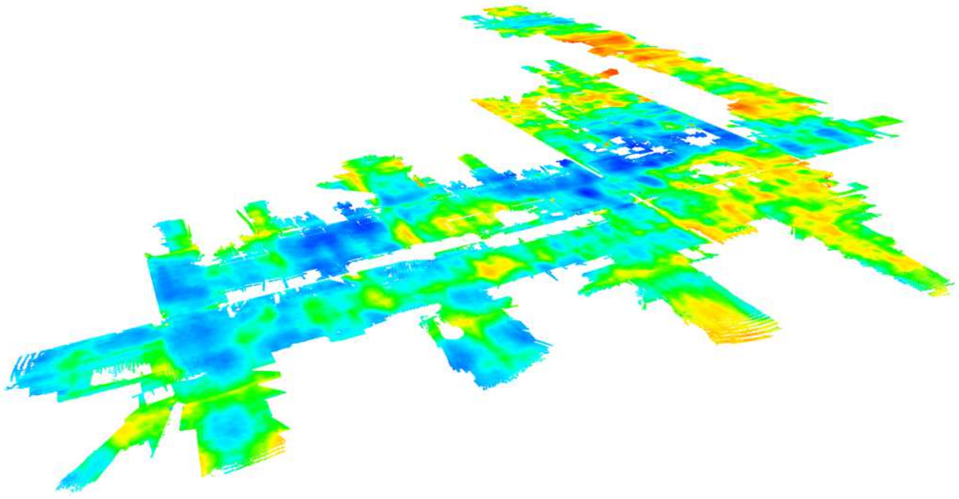


Figure 3.18: *Dataset*01 result (3D view).

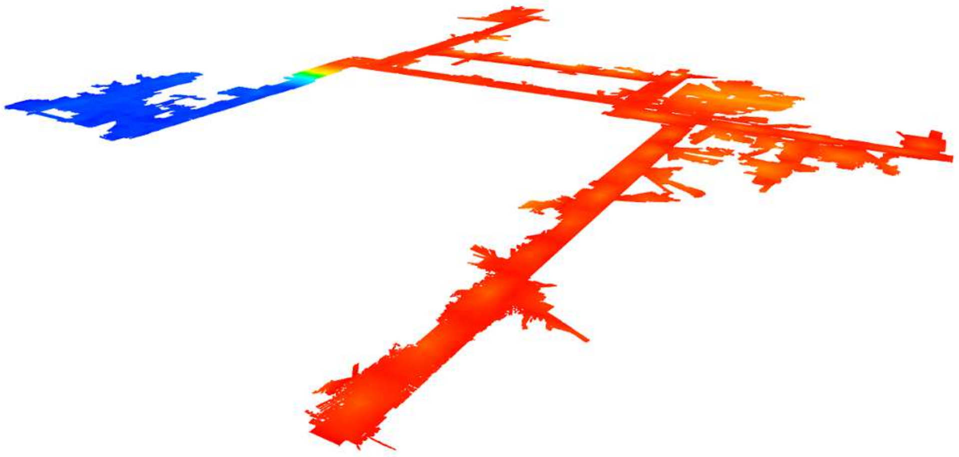


Figure 3.19: *Dataset*03 result (3D view).

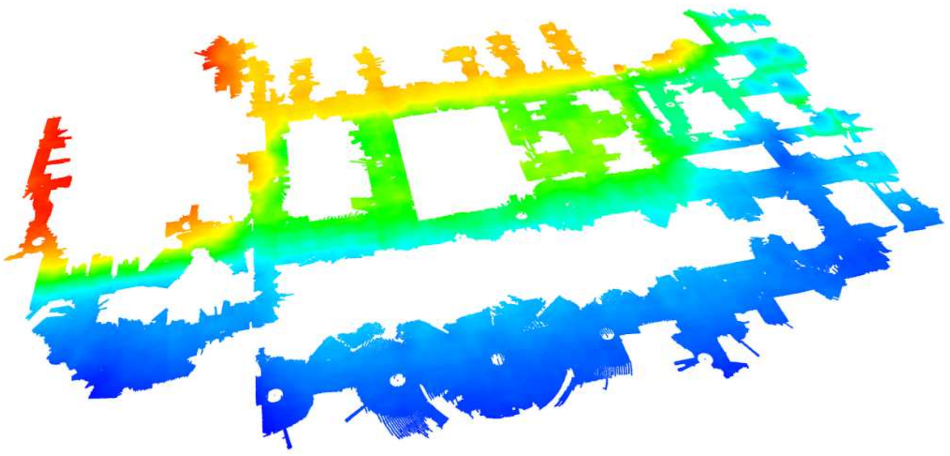


Figure 3.20: *Dataset*04 result (3D view).

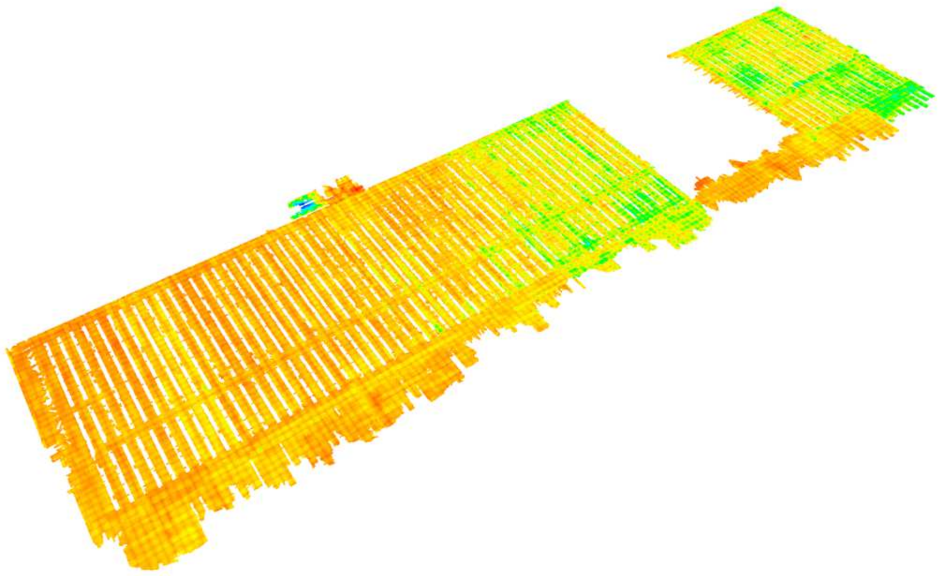


Figure 3.21: *Dataset*05 result (3D view).

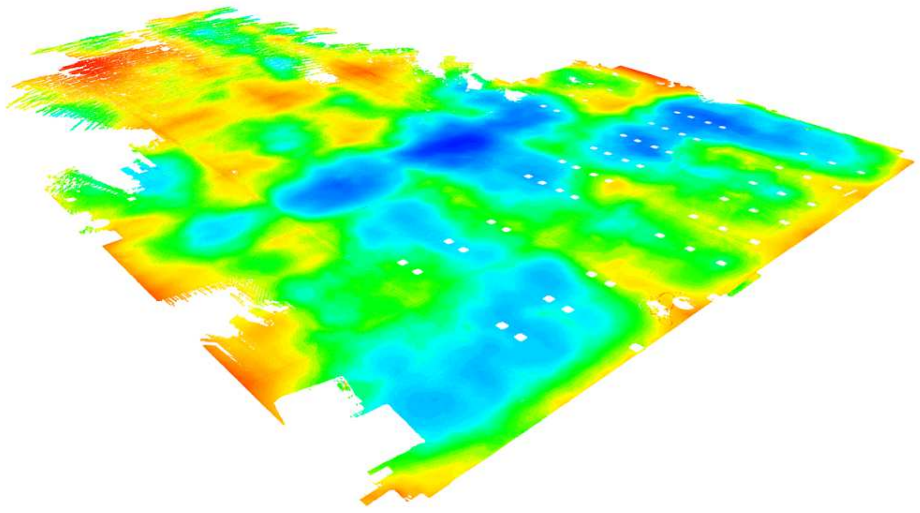


Figure 3.22: Dataset *RND* result (3D view).

[77] available within the Point Cloud Library (PCL). Evaluation of both methods was performed against the ground truth of *Dataset*00, which was manually annotated. Results, provided in Table 3.2, indicate that the proposed approach is more accurate and more efficient. Indeed, the proposed algorithm performs ground extraction on the whole dataset in 17 s at full resolution. Conversely, since PMF is not able to process the whole dataset in a reasonable amount of time, input data have been down-sampled and split into smaller parts of about 100 m$^2$. Results were then aggregated. PMF achieves a comparable processing time only when the input data are down-sampled at about $10 - 20$ cm, so it can be concluded that PMF is not applicable in large scale environments. Despite a high accuracy and precision, PMF has a considerably higher number of false positives, at each resolution, which is not acceptable in industrial applications that require a low false positive rate. Moreover, even if precision of PMF increases at lower resolution levels, this comes at the cost of an increased false negative rate.

## 3.5 Discussion

In this chapter a novel approach for ground segmentation has been proposed that can be applied in environments with non-planar regions. The method supports large scale point clouds acquired using a terrestrial laser scanner from multiple scan stations and it has been evaluated with a slope up to 2% (a high value for industrial environments). The key features are the robust estimation of ground points below each scan station, which act as seeds for region growing, and an efficient implementation based on parallel processing and caching techniques. The algorithm does not require fine tuning of any threshold. Threshold values (determined experimentally) have been kept constant across all the experiments and can be reused when working with similar sensors. Evaluation, in contrast to previous works, has been carried out in large-scale industrial scenarios containing billions of points. The proposed approach achieves better performance than a ground segmentation algorithm based on a progressive morphological filter. A video that shows the phases of the proposed approach is available.[1]

[1]`www.ce.unipr.it/~aleotti/GroundSegmentation.mp4`

# Chapter 4

# Floor Plan Generation

## 4.1 Introduction

This chapter proposes a novel approach for automatic floor plan generation of indoor environments. The method is particularly suited for large and complex buildings, like industrial, commercial, and public premises. Input data consists of a large-scale registered point cloud that is acquired from a TLS survey. The approach extracts a set of 2D polylines representing the floor plan. The main technical novelties over previous work are that the method neither assumes the presence of a flat ground nor a flat ceiling. No assumptions are made about the maximal allowed slope of the ground. Moreover, walls are neither assumed to be planar nor having orthogonal intersections. Also, the collected datasets include many cluttered areas, whose layout is included in the generated floor plan. A further contribution is that experiments were carried out in considerably larger environments than in related work. The algorithm achieves a floor plan reconstruction with the same resolution of the adopted sensor, and it was evaluated using an experimental setup that allows a resolution of 5 mm. The proposed approach detects structural elements, i.e. parts having a constant section over the entire height of the building, like walls and columns, including their internal volume. It is assumed that noisy points, due to moving objects during the TLS, are removed from the input point cloud [72]. All but one dataset were obtained in automated warehouses

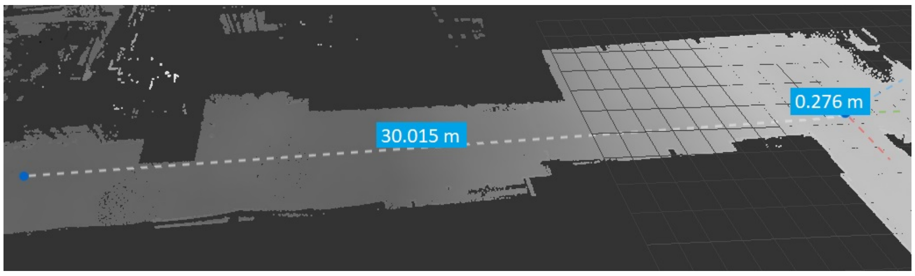


Figure 4.1: Example of ground slope in industrial scenarios. Height difference of two points (at 30.015 m distance) is 0.276 m.

from food and beverage companies. The design of such industrial environments is largely manual, and it requires the definition of all the feasible AGV paths. This is a complex and time consuming task, which is prone to errors due to outdated or even missing layout maps. Therefore, there is a growing need of automatic tools for floor plan reconstruction. Most related works on floor plan assume the presence of planar or near planar surfaces [8, 33, 3, 56, 4, 51, 68, 52, 1, 54, 50, 46, 55, 71]. In particular, the ground or the ceiling (or both) are often considered as flat. This is a strong assumption that does not hold for industrial environments (Fig. 4.1). A common strategy is to detect the ground and the ceiling by analyzing the peaks of the point cloud histogram over the z-axis [33], or to find the highest (or lowest) z coordinate [3]. Other solutions apply RANSAC [56, 4], or horizontal slicing followed by line extraction and clustering [51]. A more advanced technique is to adopt energy minimization under the Markov Random Field model [68]. The approaches in [52, 1] were the first to explicitly consider heavy cluttered environments. The work in [54] incorporates architectural priors like orthogonal intersections between walls. A method for building a semantic grid map for buildings with multiple planar floors, evaluated using simulated data, was introduced in [55]. Another approach based on floor separation and triangulation of wall samples was presented in [71]. In [39] the FloorNet framework was described that performs floor plan reconstruction from RGBD video streams, using a deep neural network trained with pre-defined features.

## 4.2 Methodology

The proposed method consists of four sequential phases. In the first phase, a low resolution voxel grid is computed to obtain a 3D representation of occupied, empty and unknown space. A segmentation algorithm [25] is also executed, with the purpose of extracting the ground from a robust estimation of the points on the floor below each laser scan station. The ground is not assumed to be flat. Indeed, most industrial environments have non-planar regions and must comply with strict regulations concerning minimum ground slope for water drainage (Fig. 4.1). In the second phase, structural elements are detected thanks to their constant section over the entire height of the building. In the third phase, a floor plan model is computed as a low resolution raster image using 2D voxel histograms. Information about structural elements is used to compensate the lack of information due to unknown voxels. The fourth phase computes the contours of occupied areas from the raster image, converting them into a set of polylines. The 2D contour map is then refined to achieve a higher resolution by upsampling the polylines, and by mapping the point samples on the 2D $(x, y)$ projection of the measured points of the scans.

### 4.2.1 Voxel grid computation and ground segmentation

In the first phase a ternary-state 3D voxel grid is computed by applying for each laser scan a ray tracing algorithm. Possible voxel states are full (F), empty (E) and unknown (U). A sub-centimetre voxel size can not be managed due to the high computational and memory consumption cost of performing ray tracing in large scale environments. As an example, in a rectangular $600 \times 300$ metres building with 15 m height ($2.7 \cdot 10^6$ m$^3$ volume) and a voxel size of 5 mm, the 3D voxel grid would contain $21.6 \cdot 10^{12}$ voxels, and the total memory usage would be about 5 Terabytes. Hence, a large voxel size $r_v$ is adopted in the early stages of the proposed pipeline. Loss of information caused by the low-resolution voxel grid is recovered in the last phase, as described in Section 4.2.4, through interpolation of the resulting 2D contour map by exploiting the original measured points of the laser scans. Furthermore, the choice of working on a low-resolution 3D voxel grid has the advantage of filtering measurement noise

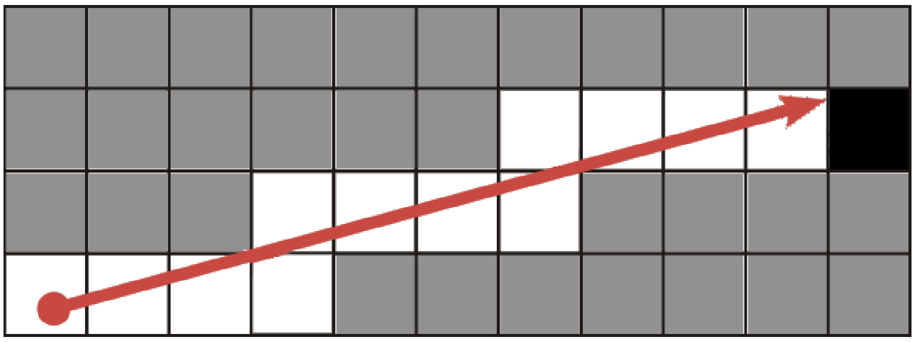


Figure 4.2: Example of ray tracing algorithm. All voxels are initially set to Unknown (grey cells). Voxels traversed by laser beams are then set to Empty (white cells). Finally, voxels containing measured points are set to Full (black cells).

in the second phase, when it comes to detecting vertically aligned occupied voxels. The minimum ($x_{min}$,$y_{min}$,$z_{min}$) and maximum ($x_{max}$,$y_{max}$,$z_{max}$) coordinates of the workspace are computed from the input point cloud. Initially all voxels are set to unknown. All the scans are then processed in parallel and unknown voxels traversed by rays are set to empty.

Each raw scan is also cleaned of points due to dynamic objects [72], i.e. object that moved during a scan, and voxels containing measured points are set to full. While performing ray tracing, the following priority order for the voxel states is established: $F \gg E \gg U$, meaning that the attribute of a voxel can not be changed to a lower priority value. Following such priority order on the voxel state enables the ray tracing algorithm to work in parallel for each ray. The proposed algorithm is also able to scale for different environment sizes and hardware resources by splitting the voxel grid into sub-grids. Each sub-grid is managed separately. Indeed, the ray tracing algorithm is able to work transparently through the sub-grids while traversing the voxel grid. The result is a tri-state voxel grid $V$ with values F, E or U for each $v_{ijk} \in V$ with $0 \leq i < W$, $0 \leq j < L$, $0 \leq k < H$ where $(W, L, H)$ are the integer boundaries computed as:

$$[W, L, H] = \left\lceil \frac{1}{r_v} \left( [x_{max}, y_{max}, z_{max}] - [x_{min}, y_{min}, z_{min}] \right) \right\rceil$$

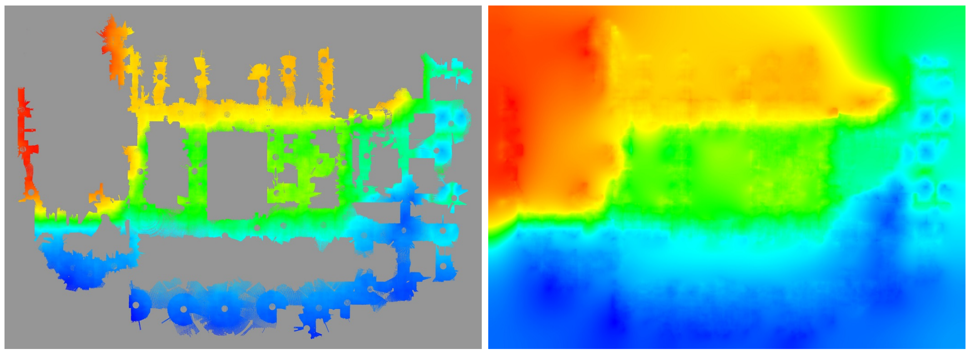


Figure 4.3: Ground segmentation (Chapter 3) of *Dataset*06 before (left) and after filling missing parts (right). Color scale represents height values ranging linearly from blue (−0.256 m) to red (0.415 m).

The voxel $(v_i, v_j, v_k)$ to which a point $(p_x, p_y, p_z)$ belongs is given by:

$$[v_i, v_j, v_k] = \left\lfloor \frac{1}{r_v} \left( [p_x, p_y, p_z] - [x_{min}, y_{min}, z_{min}] \right) \right\rfloor$$

In the first phase, the ground segmentation algorithm presented in Chapter 3 [25] is also executed, which is exploited in the subsequent phases. The output of the ground segmentation algorithm is a filled height image (Fig. 4.3), containing the local ground height in each 2D cell with size 5 mm. Ground height is represented as a 3D function $f_g(x, y)$. A point $(p_x, p_y, p_z)$ is considered as belonging to the ground if $|p_z - f_g(p_x, p_y)| < r_v$, where the voxel size $r_v$ here accounts for ground surface uncertainty.

### 4.2.2 Detection of structural elements

In the second phase, structural elements with a constant section over the entire height of the building, like walls and columns, are automatically detected. A 2D histogram $\widehat{H} \in \mathbb{N}^{W \times L}$ is computed by counting the number of unknown and full voxels in each column of the voxel grid, i.e. bin $\widehat{h}_{ij}$ of the histogram $\widehat{H}$ is given by:

$$\widehat{h}_{ij} = |\{k : v_{ijk} = F \vee v_{ijk} = U\}|$$

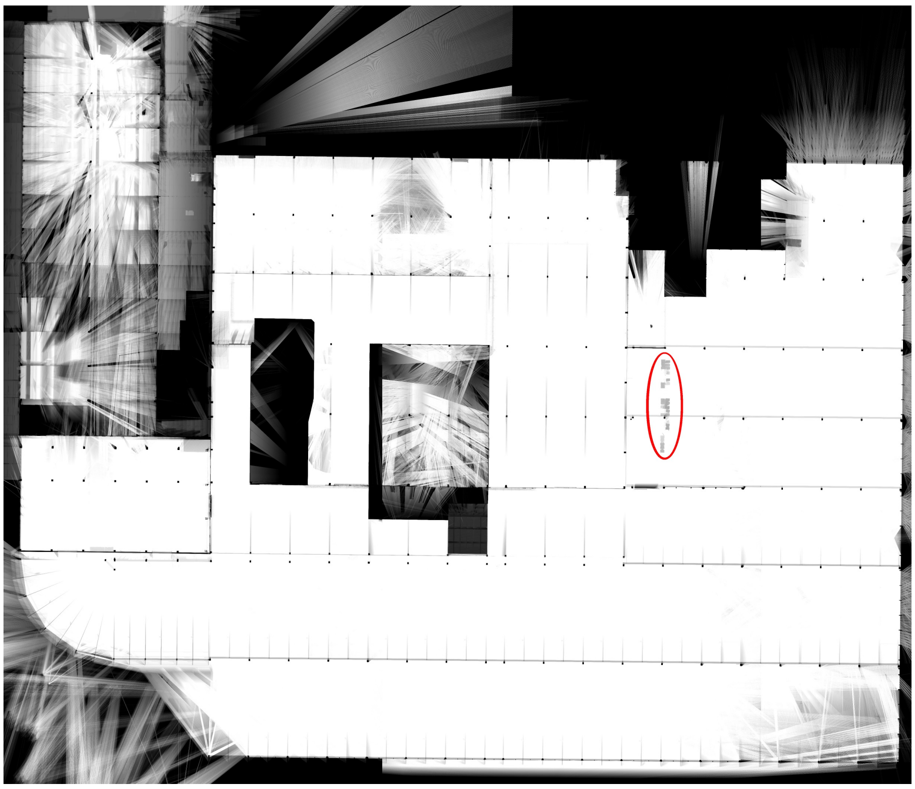


Figure 4.4: Gray scale image obtained from the 2D histogram $\widehat{H}$ of *Dataset*06. Values range from white cells, which contain only empty voxels, to black cells, that do not contain any empty voxel. The red ellipse highlights a highly cluttered unknown region (i.e. a pallet rack).

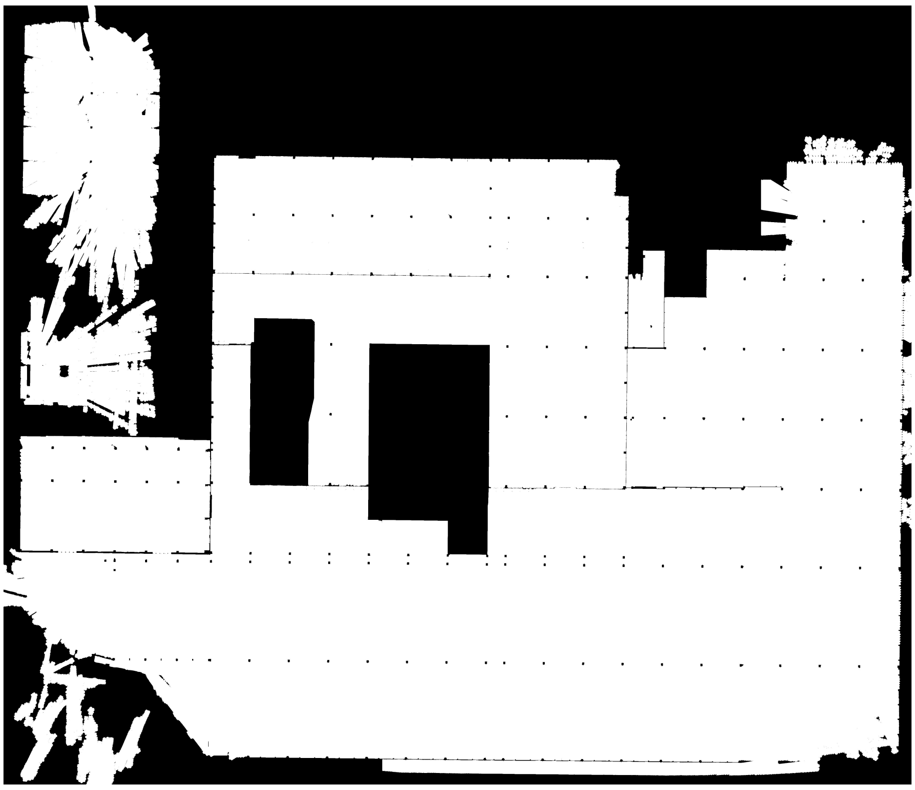


Figure 4.5: Structural elements of *Dataset*06 (displayed as black pixels with value 1, on a white background with value 0) after binarization, closing and region filling.

Full voxels are included in the count to handle borders of the structural elements. The histogram is then converted to a grey scale image with pixel size $r_v$ (Fig. 4.4) and binarized with a threshold $b_{th}$ slightly smaller than the height boundary $H$ to handle overestimated measurements due to laser beam reflection. Indeed, overestimation generates missing points on scanned surfaces. Then, a morphological closing operation is performed so that gaps smaller than $\kappa$ are closed. Finally, all empty regions smaller than a threshold $r_{th}$ are removed by applying a region filling algorithm [17]. For example, threshold $r_{th}$ can be used to fill small isolated rooms that are of no interest. If all contours must be reconstructed, parameter $r_{th}$ can be set to a small value or zero.

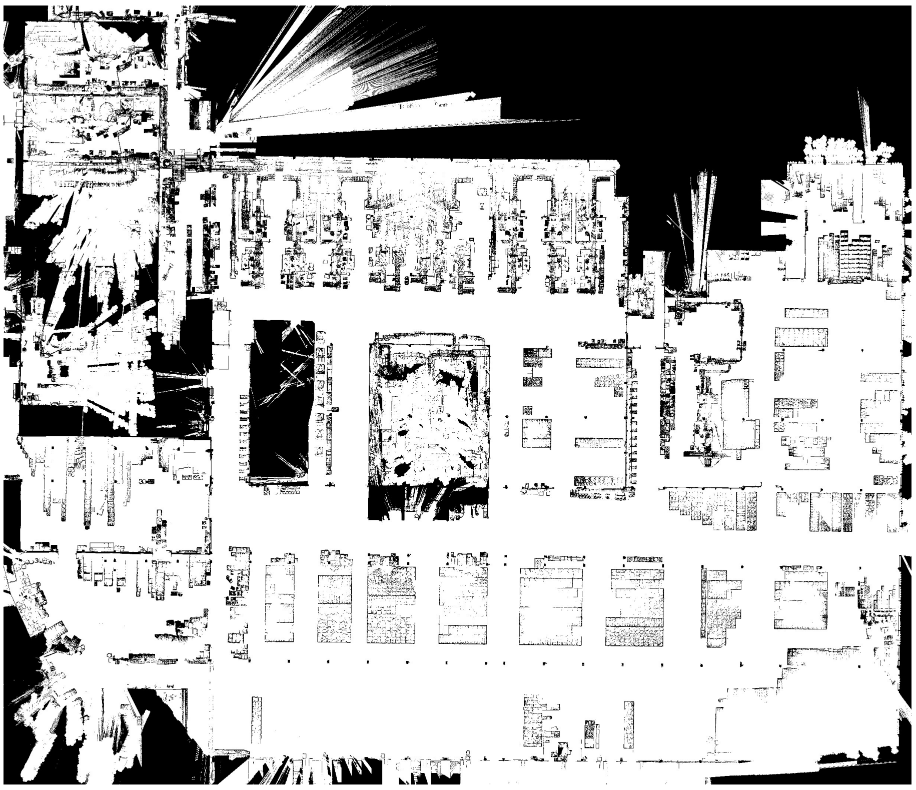


Figure 4.6: Floor plan image of *Dataset*06 before morphological closing and region filling.

The result is a binary grid $S \in \mathbb{N}^{W \times L}$, where element $s_{ij} \in S$ has the following format

$$s_{ij} = \begin{cases} 1 & \text{if cell } (i, j) \text{ contains a structural element} \\ 0 & \text{otherwise} \end{cases}$$

Fig. 4.5 shows the binary image of the obtained structural elements. Regions within small and isolated rooms are also considered as structural elements.

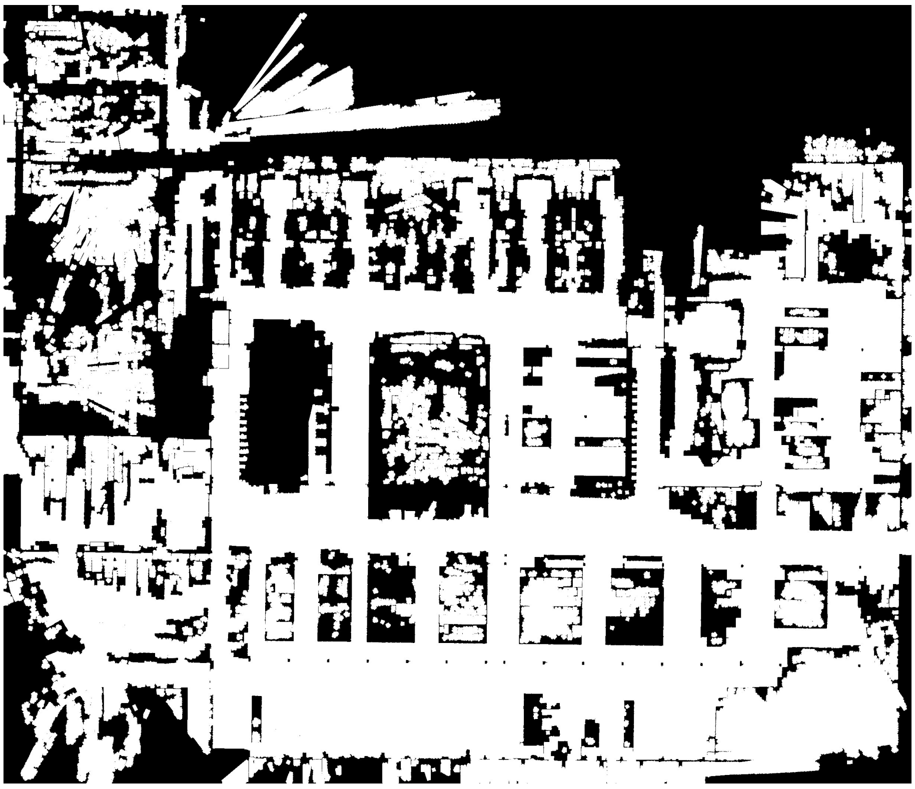


Figure 4.7: Floor plan image of *Dataset*06 after morphological closing.

### 4.2.3 Low resolution floor plan image generation

In the third phase a low resolution image of the floor plan is generated by exploiting both the voxel grid and the structural elements. The algorithm can work up to a fixed height limit $\Delta_h$ with respect the ground level of each 2D cell. Only the elements of the environment below this limit are inserted in the final floor plan. In particular, the algorithm only considers voxels centered at $v_{xyz} = (v_x, v_y, v_z)$, that satisfy the following condition:

$$f_g(v_x, v_y) + r_v \leq v_z \leq f_g(v_x, v_y) + \Delta_h \tag{4.1}$$

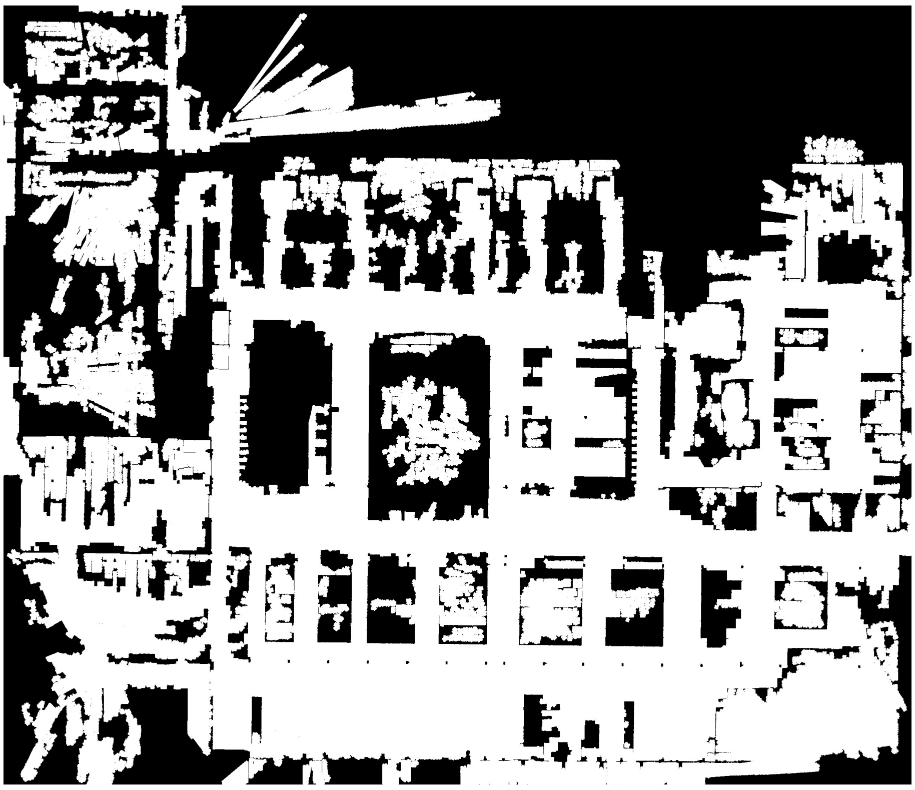


Figure 4.8: Floor plan imageof *Dataset*06 after region filling.

where $f_g(x, y)$ is the ground height. The inequality on the left ensures that only the voxels above the ground are considered for each cell, while the inequality on the right defines the height limit with respect to the ground level of each cell. The height limit is optional and can be customized on the basis of the application. A histogram $\widetilde{H} \in \mathbb{N}^{W \times L}$ is created by counting empty voxels for each column below the height limit. Each bin $\widetilde{h}_{ij}$ of $\widetilde{H}$ is therefore computed as:

$$\widetilde{h}_{ij} = |\{k : v_{ijk} = E\}|$$

Table 4.1: Datasets used in the experimental evaluation.

| **Dataset** | **Scans** | **Res. [mrad]** | **Points** | **Size [m$^2$]** |
|---|---:|---:|---:|---:|
| *Dataset*01 | 11 | 1.2 | 106 417 059 | 788 |
| *Dataset*02 | 48 | 0.6 | 1 441 032 861 | 17 675 |
| *Dataset*03 | 54 | 1.2 | 497 805 241 | 43 000 |
| *Dataset*04 | 58 | 1.2 | 544 864 429 | 18 396 |
| *Dataset*05 | 63 | 1.2 | 537 812 076 | 63 317 |
| *Dataset*06 | 117 | 1.2 | 1 031 886 921 | 51 450 |

A grid $M \in \mathbb{N}^{W \times L}$ is also generated that marks the cells that contain at least one full voxel in their voxel column, i.e. elements $m_{ij}$ of $M$ are defined as:

$$m_{ij} = \begin{cases} 1 & \text{if } \exists k : v_{ijk} = F \\ 0 & \text{otherwise} \end{cases}$$

A 2D image $P$ is then generated by exploiting the grid $M$, the histogram of empty voxels $\widetilde{H}$ as well as information about structural elements $S$. Elements $p_{ij}$ of the binary image $P$ are computed as:

$$p_{ij} = \begin{cases} 1 & \text{if } m_{ij} = 1 \\ 0 & \text{if } m_{ij} = 0 \wedge \widetilde{h}_{ij} \geq e_{th} \left\lfloor \frac{\Delta_h - r_v}{r_v} \right\rfloor \\ s_{ij} & \text{otherwise} \end{cases} \tag{4.2}$$

A single full voxel below the height limit is sufficient to consider a cell as full. Such conservative policy is important to avoid missing any object or obstacle on the floor plan. Threshold $e_{th}$, with $0 \leq e_{th} \leq 1$, is used to determine when a voxel column, that was not labeled as full, contains enough empty voxels so that the corresponding cell can be considered as empty. If a cell is neither marked as full nor tagged as empty from the first two conditions in (4.2), it means that its voxel column does not contain any full voxel and that it contains a large number of unknown voxels like, for example, in strongly occluded regions (Fig. 4.9). In this case, structural elements computed in the second phase are exploited to compensate for missing information due to unknown

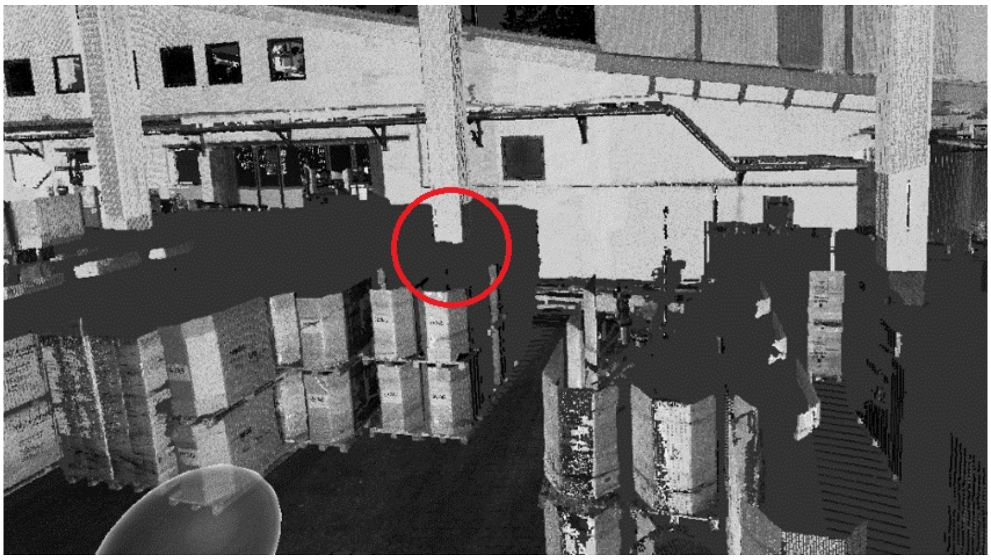


Figure 4.9: Example of a column (highlighted by the red circle) of *Dataset*06 whose lower part is highly occluded due to clutter. As the volume occupied by the column is detected as a structural element, the proposed algorithm is able to reconstruct the layout of the column.

regions of space, by setting the cell value to $s_{ij}$. An example of image $P$ is shown in Fig. 4.6. The same morphological closing procedure described in Section 4.2.2 is then applied to connect regions that are closer than $\kappa$ (Fig. 4.7) and, after filling all the gaps smaller than $r_{th}$, the output of the algorithm is the low-resolution image $P$ that depicts the floor plan (Fig. 4.8).

### 4.2.4 Contours extraction and refinement

In the last phase 2D contours are extracted from the low resolution image $P$. The set of contours is then refined at a higher resolution (5 mm), by exploiting the 2D projection of the measured points of the laser scans. Let $A$ be the whole point cloud, and $\widetilde{A} \subset A$ a sub-set that contains the measured points belonging to voxels that satisfy (4.1), i.e.

$$\widetilde{A} = \{p \in A : f_g(p_x, p_y) + r_v \leq p_z \leq f_g(p_x, p_y) + \Delta_h\}.$$

Table 4.2: Computation time of each phase of the proposed approach.

| Dataset | Phase 1 (4.2.1) | Phase 2 (4.2.2) | Phase 3 (4.2.3) | Phase 4 (4.2.4) |
|---|---:|---:|---:|---:|
| *Dataset*01 | 194 | 3 | 2 | 4 |
| *Dataset*02 | 1987 | 24 | 43 | 117 |
| *Dataset*03 | 1065 | 203 | 122 | 36 |
| *Dataset*04 | 1211 | 49 | 63 | 38 |
| *Dataset*05 | 2836 | 149 | 279 | 98 |
| *Dataset*06 | 2549 | 167 | 318 | 181 |

Let also define $A^{\perp}$ as the set of points in $\mathbb{R}^2$ obtained by projecting points in $\widetilde{A}$ on the $z = 0$ plane. The binary image $P$ is first processed to extract the borders of objects and other elements of the environment. Contour extraction is performed by using the OpenCV library. The result is a set of polylines. Polylines are converted to world coordinates and up-sampled by splitting them into shorter segments of 5 mm length. Each vertex $b = (b_x, b_y)$ of the segments is then mapped to a vertex $b'$ as follows

$$b' = \begin{cases} \arg\min\limits_{p \in A^{\perp}} ||p - b||_2 \text{ if } \exists p \in A^{\perp} : ||p - b||_2 \leq \frac{3\sqrt{2}}{2} r_v \\ b \text{ otherwise} \end{cases} \tag{4.3}$$

where the constant threshold $\frac{3\sqrt{2}}{2} r_v$ is the maximum admissible distance from the central point in a $3 \times 3$ neighborhood, i.e. one and a half the diagonal of the image pixel. That is, if the distance between vertex $b$ and the closest point in $A^{\perp}$ is lower than $\frac{3\sqrt{2}}{2} r_v$, then $b$ is mapped to the closest point. Otherwise, vertex $b$ is not changed. An example of the refinement operation is shown in Fig. 4.10, while the final floor plan is shown in Fig. 4.11. A video that clarifies the refinement procedure is also available.[1]

## 4.3 Experimental evaluation

Processing was performed on a desktop Intel i7-4790 (3.60 GHz), with 32 GB RAM. Table 4.1 reports information about the datasets and the computation time of each

[1]www.ce.unipr.it/~aleotti/LayoutRefinement.mp4

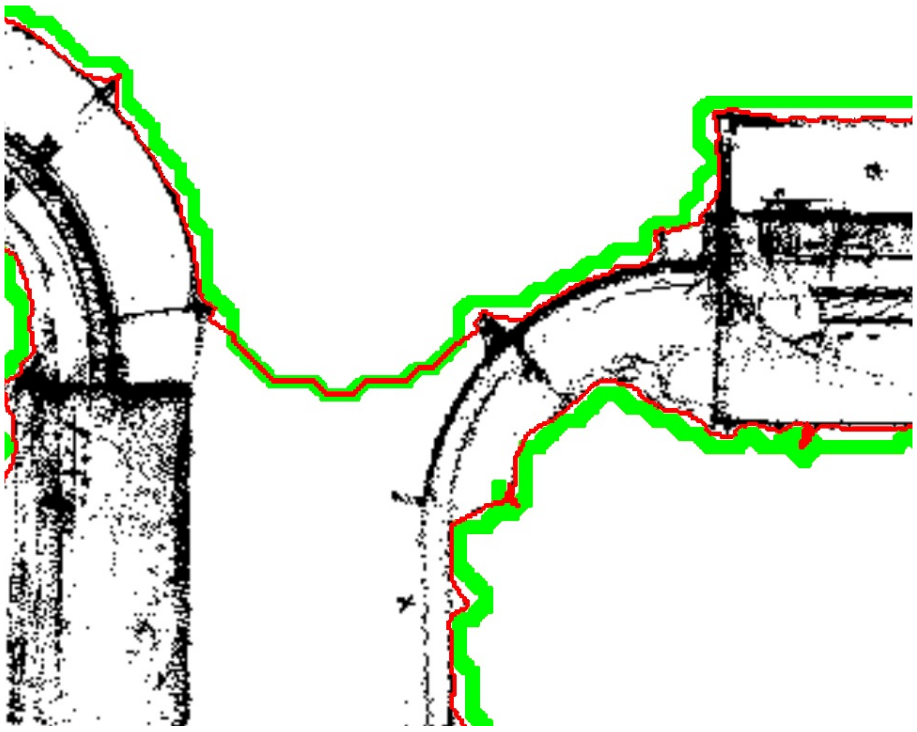


Figure 4.10: Example of polylines refinement. Black dots are points in $A^{\perp}$ (Section 4.2.4), green polylines are the contours obtained from the low resolution image, red polylines are the contours after refinement. It can be noticed that red lines better fit the object contours.

phase. The plant size ranged from $10^3$ m$^2$ (*Dataset*01) to almost $10^6$ m$^2$ (*Dataset*05). The number of scans ranged from 11 to 117, containing a total number of points from about $10^8$ to over $10^9$. Most of the TLS surveys have been performed at 1.2 mrad resolution. *Dataset*02 was obtained using a scan resolution of 0.6 mrad. All industrial datasets contain production lines and a large number of racks for pallet storage. *Dataset*02 is a University building, thus showing the capability of the algorithm to work in different indoor environments. Parameters used in the experiments were set as follows: $b_{th} = 0.9H$, $\kappa = 20$ cm, $r_{th} = 100$ m$^2$, $e_{th} = 0.5$ (meaning that a

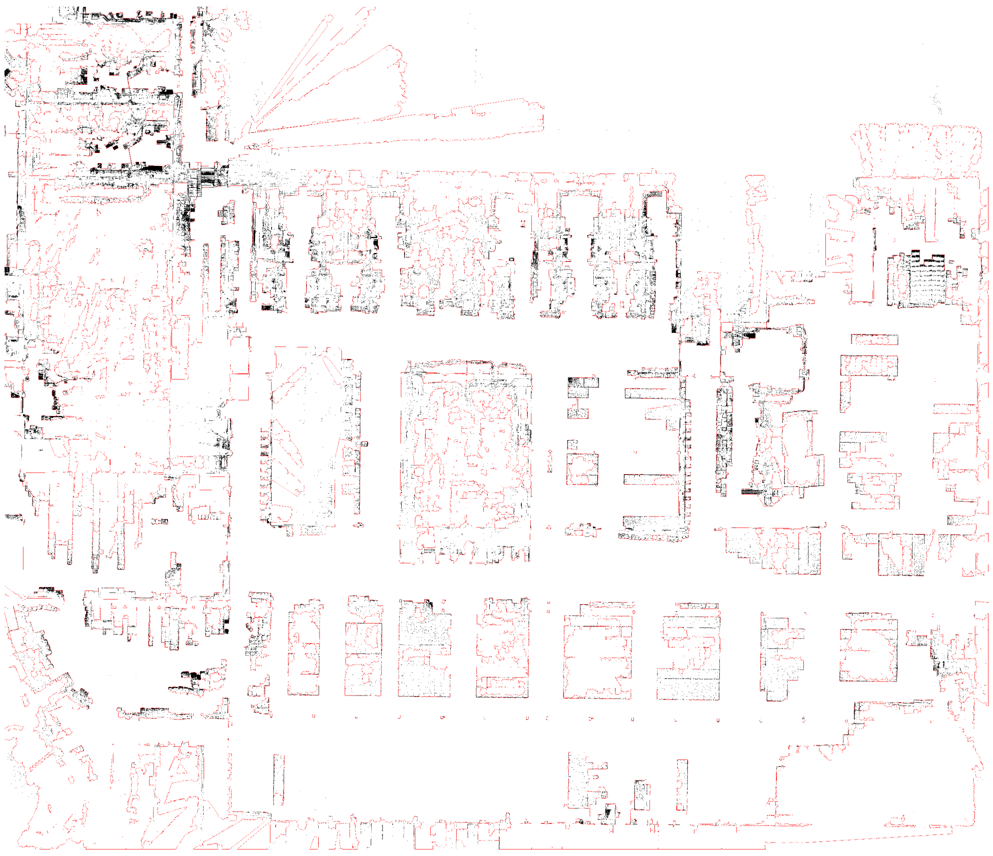


Figure 4.11: Floor plan of *Dataset*06 with $r_v = 5$ cm. Black dots are points in $A^{\perp}$ (Section 4.2.4). Some artifacts are visible in non-critical regions where few scans were performed or outside the perimeter walls..

voxel column should at least have 50% of empty voxels to be considered as empty). Threshold $r_{th}$ has been lowered to 16 m$^2$ in *Dataset*02 to preserve reconstruction of small rooms. Height limit $\Delta_h$ was set equal to the height of the vehicles operating in the warehouse in the industrial datasets, while $\Delta_h$ was set lower than the height of the doors in *Dataset*02. Profiling shows that most time is spent in the first phase due to ray tracing. Figures 4.11, 4.12, 4.13 and 4.14 show the floor plans ($r_v = 5$ cm) of *Dataset*06, *Dataset*01, *Dataset*02, and *Dataset*04 respectively. The generated

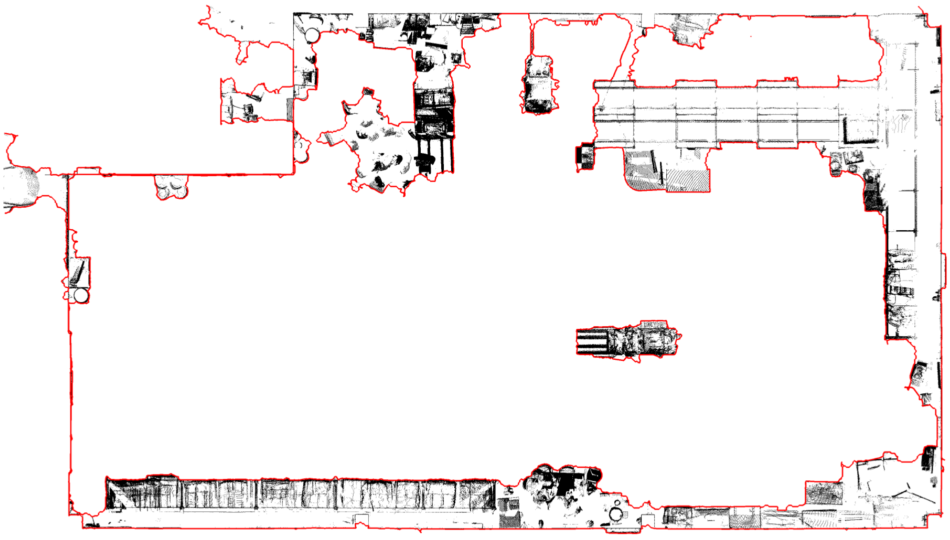


Figure 4.12: Floor plan of *Dataset*01 with $r_v = 5$ cm.

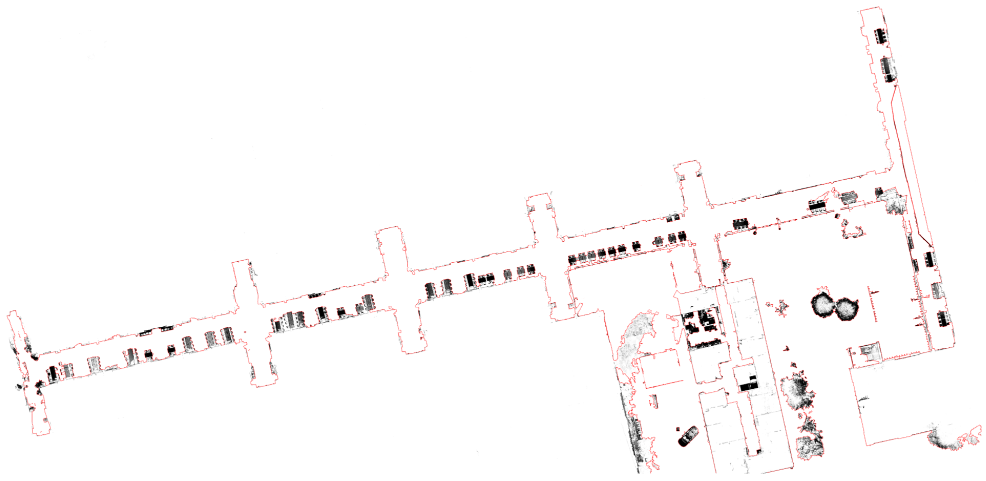


Figure 4.13: Floor plan of *Dataset*02 with $r_v = 5$ cm.

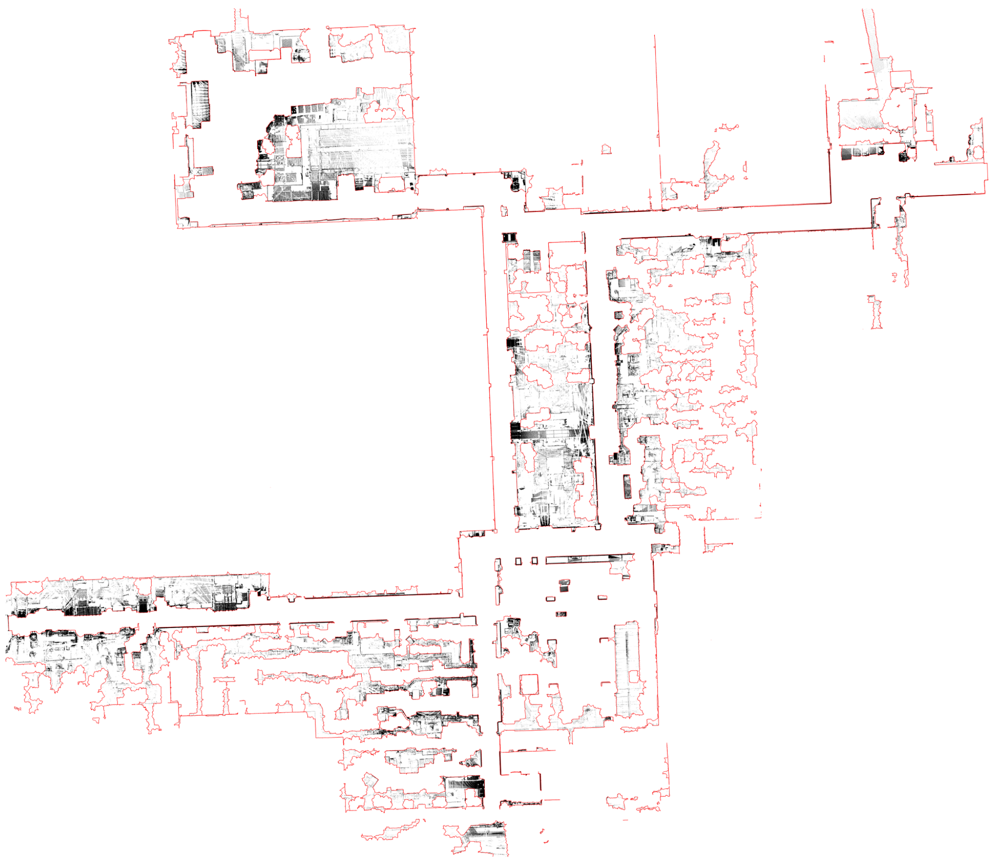


Figure 4.14: Floor plan of *Dataset*04 with $r_v = 5$ cm.

polylines (red lines) follow the contours of walls and other cluttered regions as well as the borders of the structural elements, like columns. The height histogram of *Dataset*02 shows that both the ground and the ceiling can not be clearly identified, therefore, methods that assume the presence of planar surfaces would not be applicable. Thus, two simpler variants of the proposed algorithm were considered for comparison. The first algorithm used for comparison does not require computation of the voxel grid and it does not perform detection of structural elements. In particular, the algorithm is a less conservative approach that considers all unknown space as empty by defining

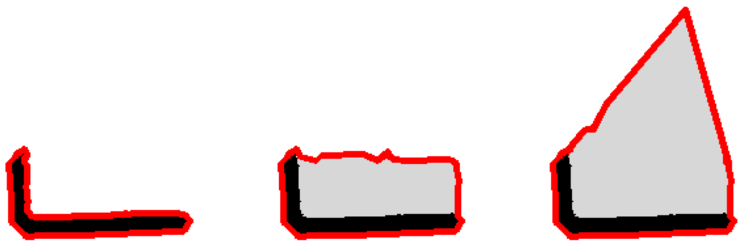


Figure 4.15: Example of the reconstructed layout of a column from *Dataset*04. Unknown space is colored in grey, while occupied space is displayed in black. Only the proposed algorithm (center) achieves a correct reconstruction. The two simplified algorithms that consider all unknown space as empty (left) and all unknown space as occupied (right) fail to extract the shape of the column.

$p_{ij}$, in equation (4.2), as $p_{ij} = m_{ij}$ The main advantage of this approach is that the total computational time is greatly reduced by about $80\% - 90\%$. Conversely, the second algorithm used for comparison is a more conservative approach that considers unknown space as occupied. This solution requires computation of the voxel grid and it does not perform structural element detection. In this algorithm the value of $p_{ij}$ is defined as:

$$p_{ij} = \begin{cases} 1 & \text{if } m_{ij} = 1 \vee \widetilde{h}_{ij} < e_{th} \left\lfloor \frac{\Delta_h - r_v}{r_v} \right\rfloor \\ 0 & \text{otherwise} \end{cases}$$

Figure 4.15 shows an example layout of a highly occluded column from *Dataset*04 by using the proposed approach, as well as by using the two simplified methods. Only the proposed method achieves a correct reconstruction of the layout of the column, thanks to the detection of structural elements. Indeed, while the lower part of the column close to the ground was highly occluded, the upper part was properly scanned, and that was sufficient to detect the column as a structural element. The two simpler methods fail to recover the shape of the column as they are both highly dependent on laser occlusions. Figure 4.16 shows that the total computation time is inversely proportional to the voxel size $r_v$. A quantitative evaluation of the floor plan quality

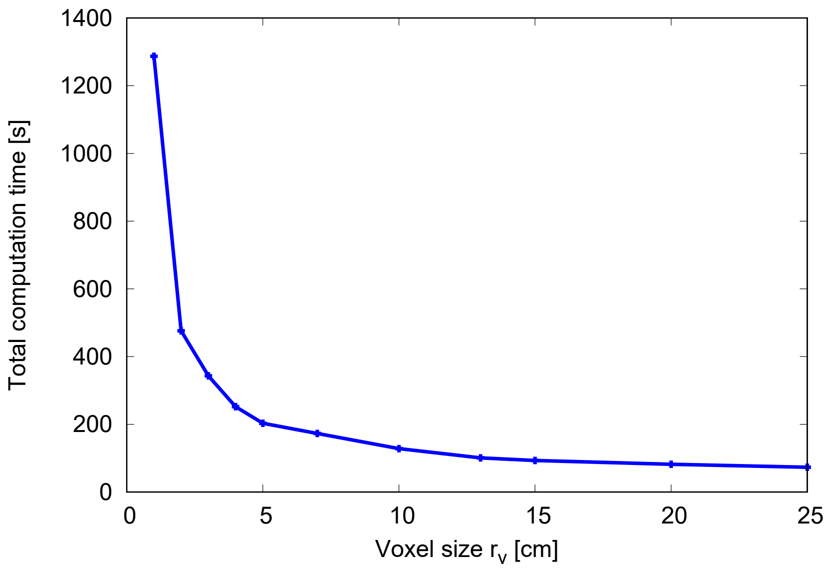


Figure 4.16: Total computation time with respect to $r_v$ in $Dataset01$.

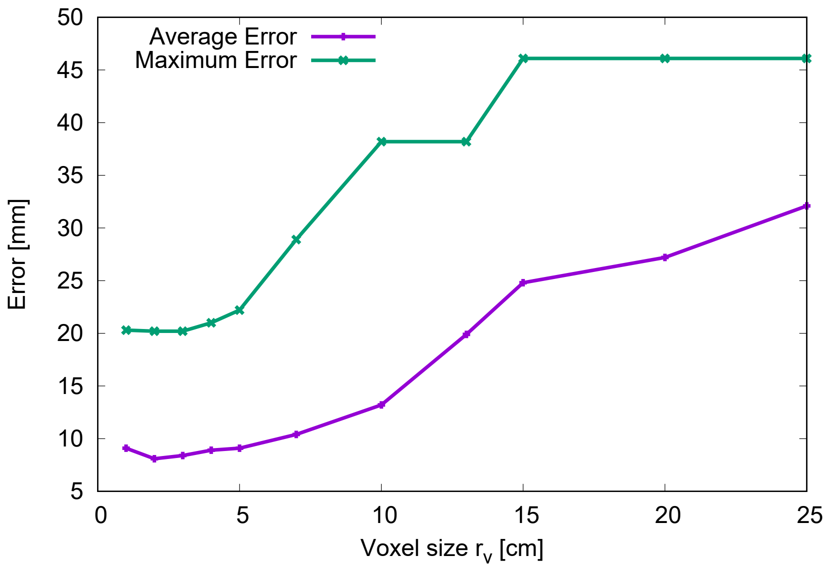


Figure 4.17: Floor plan reconstruction error with respect to $r_v$ in $Dataset01$.

has also been carried out against a ground truth dataset obtained by using the Leica Nova TS60i 0.5″ R1000 total station. A total of 27 key points from *Dataset*01 were acquired, located on wall corners and rack edges. Figure 4.17 also shows the average and maximum error between the ground truth key points and the corresponding vertices of the generated polylines, with respect to the voxel size $r_v$. At low values of $r_v$ the error is close to the maximal resolution of the TLS. The reconstruction error is about 1 cm even when starting from a higher voxel size, e.g. $r_v = 5$ cm, which is required in environments considerably larger than *Dataset*01, such as *Dataset*05 and *Dataset*06, thus confirming the effectiveness of the refinement phase.

## 4.4 Discussion

A novel method to extract the floor plan of an indoor environment was presented with an accuracy comparable to the resolution of the TLS. In contrast to previous work the algorithm does not assume the existence of planar surfaces. Therefore, the algorithm can be applied in complex industrial plants. Experiments were performed in large scale point clouds containing over $10^9$ points, acquired by a TLS. The method is based on the computation of a 2D histogram of voxels, and on finding structural elements that have a constant section over the entire height of the building. A limitation is that the approach can fail to reconstruct the contour of partially occluded structures with non-constant section like arches.

# Chapter 5

# 3D Warehouse Visualization and Collision Checking

## 5.1 Introduction

The use of VR technologies has strong potential to shorten the design and development time of complex industrial environments [43, 70].

In this chapter, an immersive and interactive desktop VR system is presented for visualization of AGVs moving along specified paths in warehouses. The main novelty of the approach is that the VR system enables visualization of large-scale industrial datasets with real-time collision checking. In particular, the VR system supports visualization of large scale point clouds acquired from a Terrestrial Laser Scanning survey. TLS allows generation of massive colored point clouds with high accuracy, that can be displayed by using point based rendering techniques to achieve photorealistic visualization [67, 7, 38].

AGV animation is obtained by taking into account the local slope of the ground, which is computed from the point cloud model of the environment. Visualization of AGV safety zones is also supported. Moreover, the VR system enables real-time collision detection (at 60 fps) between the AGV model and the static point cloud model of the environment. Collision checking is important to evaluate the feasibility of AGV paths, that usually are manually configured by the warehouse designer.

The closest work is by Eilers et al. [20], where a VR system was developed to visualize data in real-time from a real world facility and to measure performance. However, the system did not support collision detection. Non-VR 3D computer simulations were also proposed for AGV systems performing logistics tasks in [48, 74, 28] without collision detection.

VR applications of point cloud visualization and interaction in non-industrial environments were presented in [12, 40, 69]. Bruder et al. [12] introduced a human-robot telepresence setup with point cloud rendering. In [40] an immersive bi-manual user interface was proposed for interaction, selection and annotation of point cloud data. In [69] a method was proposed for accurate rendering of LiDAR data.

Accessibility analysis in industrial environments has been studied by Shellshear et al. [62], where an algorithm was proposed to determine the maximum size of an object that can move along a fixed trajectory of an assembly line without collisions. The approach was based on an octree volumetric representation of the environment.

The problem of collision detection with point cloud data has been addressed in several previous works. However, previous approaches did non focus on collision detection of AGV in warehouses and did not support real-time visualization in VR [53, 22, 60, 49, 61]. The work in [53] addressed the problem of collision detection and distance computation on point cloud data obtained from the robot sensors. In [22] a method was presented for massive point cloud collision detection that simplifies the input data according to a maximal allowed error for shortest distance queries. Schauer et al. [60] investigated algorithms for large scale point cloud to point cloud collision detection on CPU and GPU. In [49] an efficient collision detection approach was developed, based on an angle-space depth map, that supports large-scale point clouds acquired using a TLS. In [61] an algorithm similar to [62] was investigated for computation of maximum collision-free volumes in large scale point clouds. In [59] an approach for collision detection between point clouds was proposed that implements a highly optimized k-d tree and two methods for penetration depth calculation. The chapter is organized as follows. Section 5.2 presents the input data for the VR system, including the vehicle 3D model, its safety zones, the point cloud data structure used to represent the environment, the ground model and the description of vehicle paths. Section 5.3 describes the VR environment including the adopted techniques for point cloud visualization and vehicle animation. Section 5.4 discusses the algorithm for efficient collision detection between the vehicle model and the large scale point cloud. Section 5.5 reports the experiments that were carried out to assess the performance of the VR system. Section 5.6 concludes the chapter and indicates future research.

## 5.2 Input data

### 5.2.1 Vehicle model and safety zones

The AGVs that we consider in this work are equipped with planar laser scanners, mounted at the top and at the base of the vehicles. The planar laser scanner located at the top of a vehicle is used to detect reflective, pre-mapped, landmarks placed in the environment for navigation and self-localization. Planar laser scanners at the bottom of a vehicle are used for safety issues. Indeed, a number of safety zones are defined around the vehicle. If a laser sensor detects an obstacle within a safety zone the vehicle immediately stops. Figure 5.1 shows a real vehicle. The complete 3D model of the AGV, which is used for rendering, is shown in Figure 5.2, while Figure 5.3 shows a detail of the safety planar laser scanner at the bottom of the vehicle.

To achieve efficient collision detection a different 3D model is used based on a bounding box hierarchy, computed as follows. The initial model is simplified, due to the high number of vertices, and decomposed into convex parts by applying the Volumetric-Hierarchical Approximate Convex Decomposition (V-HACD 2.0) library, which extends [41] (the resulting convex decomposition is shown in Fig. 5.4). An oriented bounding box tree (OBB tree) is then created by applying a bottom-up agglomerative construction method. To build the OBB tree, the OBB of each part of the AGV convex decomposition is first computed. These volumes form the leaf nodes of the OBB tree. Then, starting from the leaves, two nodes of the OBB tree are iteratively selected and merged into a parent node. Each parent node contains the OBB enclosing the two OBBs of the children, as well as the two pointers to the children

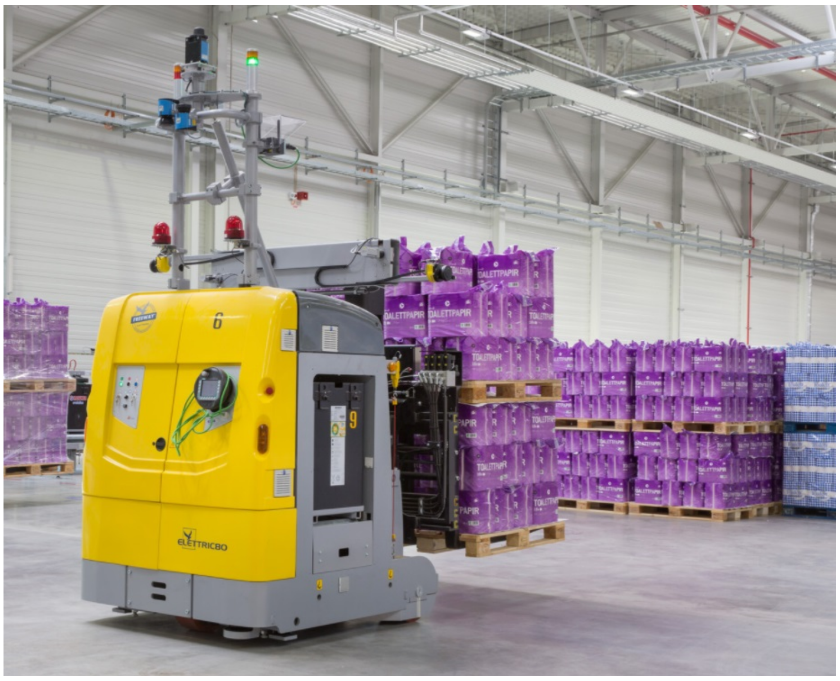


Figure 5.1: A real AGV vehicle.

nodes. The merging criterion selects the two closest nodes $(b_i^*, b_j^*)$, so that

$$(b_i^*, b_j^*) = \underset{b_i \in B, b_j \in B}{\arg\min} \, ||O(b_i) - O(b_j)||_2 \tag{5.1}$$

where $O(b_i)$ is the center of the OBB of node $b_i$, whereas $O(b_j)$ is the center of the OBB of node $b_j$. Since there are no guarantees that the obtained OBB tree will be well balanced the result must be checked for adequacy.

AGV safety zones are defined as thin volumes (about 5 cm thick) with a star shaped cross section. Figure 5.5 shows a safety zone (displayed in green color) around the vehicle model. Safety zones can change for each segment of the path as described in Section 5.2.4. Finally, the proposed system supports simultaneous simulation of multiple vehicles moving along different paths. Each vehicle can have its own shape.

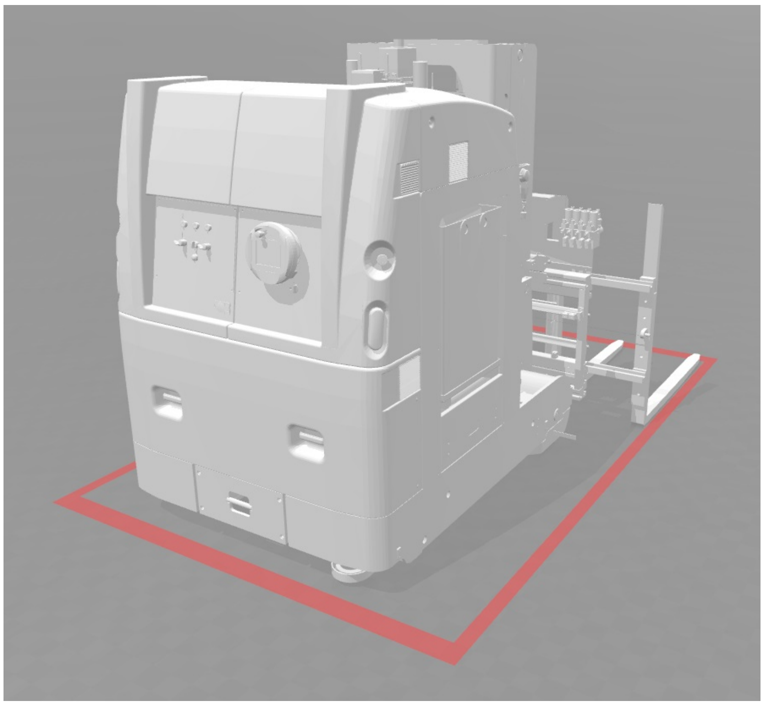


Figure 5.2: 3D vehicle model ($\backsim$ 2 million vertices).

### 5.2.2 Environment model

The static environment is described as a set of registered point clouds, obtained from a Terrestrial Laser Scanning survey. Accurate point cloud registration is performed using artificial targets, automatically detected by the scanner, that are also mapped in advance with a total station. The total amount of points is about $10^{10}$ in the reported experiments.

The registered point cloud $P$ is clustered in a regular grid of size $s$=10 m, to enable real-time collision detection in the VR simulation. Each point $v^k \in P$, with $v^k$=$[v_x^k, v_y^k, v_z^k]$, is assigned to a cluster $d_{ij} \in D^{N\times M}$ with

$$d_{ij} = \left\{ v^k \in P : \begin{array}{c} i < \frac{v_x^k - x_m}{s} < i+1 \\ j < \frac{v_y^k - y_m}{s} < j+1 \end{array} \right\} \tag{5.2}$$

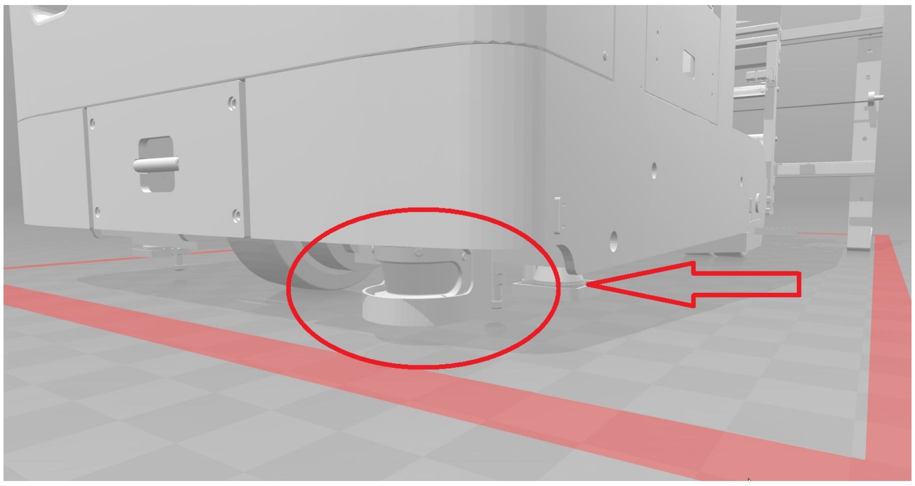


Figure 5.3: Safety planar laser scanner at the bottom of the vehicle (highlighted by a red circle).

where $x_m$ and $y_m$ are the minimum coordinates of $P$, so that $\forall v^k \in P\ x_m < v^k_x$ and $\forall v^k \in P\ y_m < v^k_y$. The clustering operation has a complexity of $O(n)$ with $n = |P|$ and it is run in parallel.

### 5.2.3 Ground model

The ground model is described by a function $z = f_g(x, y)$ that returns the local height $z$ of the ground at location $(x, y)$. Function $f_g(x, y)$ is obtained by applying the method proposed in Chapter 3, that has been successfully tested in large scale point clouds of industrial environments. The approach is based on the detection of laser scanned points located on the ground below each scan station. A fully connected graph that connects these points is then generated. Inconsistent nodes due to excessive slope are removed and, afterwards, a volume of interest that contains the ground is created. A further filtering step is performed that checks for inconsistent normal vectors. Finally, remaining nodes of the graph are used as seeds for a region growing procedure, followed by gap filling.

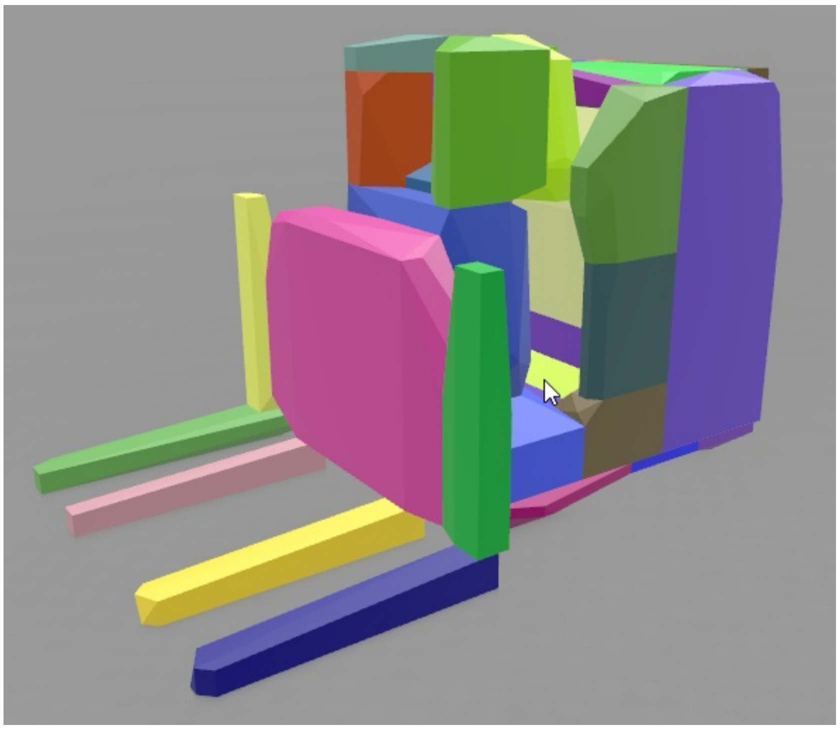


Figure 5.4: 3D vehicle model after mesh simplification and convex decomposition into 64 parts ($\backsim$ 1000 vertices). Each part is displayed with a different color.

### 5.2.4 Vehicle paths

Vehicle paths are predefined in advance as a sequence of 2D segments using a CAD program. Vehicle paths are shared with all vehicles. Each segment can be either linear or curvilinear. Linear segments are defined by a starting and an ending point, while curvilinear segments are described as a set of sampled points. Point samples $[x, y] \in \mathbb{R}^2$ are converted to 3D vertices $[x, y, f_g(x, y)] \in \mathbb{R}^3$ by adding the z-component from the ground model. Each segment is associated to a safety zone whose shape is defined by the designer according to the surrounding environment. Segments can be traveled both in forward and in backward direction.

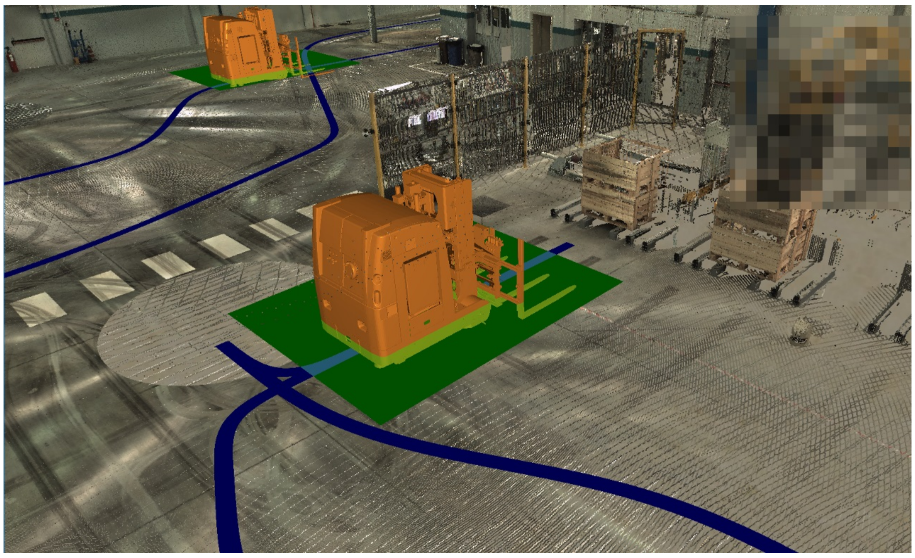


Figure 5.5: Example of the 3D virtual environment. The system supports multivehicle simulation. Paths are displayed in blue, safety zones around the vehicles are displayed in green (meaning that the safety zones are not colliding with the environment).

## 5.3 Visualization

The VR system consists of a desktop application. Visualization is based on the Oculus Rift CV1 head-mounted display. The software was developed using the Oculus Rift SDK and OpenGL. A multithreading architecture on the CPU is exploited for dynamic loading of point clusters using parallel and asynchronous tasks.

The user can interact with the VR system through the keyboard, which is used for moving in the VR environment. The position of the virtual camera is maintained at a fixed height $h_c$ from the ground, i.e. given the current 2D position of the user in the VR environment $[u_x, u_y]$, the camera position $U \in \mathbb{R}^3$ is set as

$$U = [u_x, u_y, f_g(u_x, u_y) + h_c] \tag{5.3}$$

A free-fly camera can also be enabled, if required. The user can also stop and resume vehicle animation and change the size of the rendered points of the static environment.

### 5.3.1 Vehicle path following

Vehicle animation is performed at constant speed and it is based on a purely geometric approach that does not include a dynamic model. Each vehicle is considered as a rigid body with 6 degrees of freedom. To achieve 3D animation at constant speed the vehicle at each iteration is moved along consecutive segments for a distance proportional to the time elapsed since the last frame. Segments are traveled by evaluating the sampled vertices on the path. Vehicle orientation is computed so that the vehicle heading direction is along the line connecting the current position to the next sampled vertex of the path. Vehicle's up vector is aligned with the local normal of the ground. Figure 5.6 shows a scheme that describes the animation procedure.

Vehicle paths are rendered on the ground model as OpenGL quad strips, defined as follows. For each sampled point $p_i$ of a segment, the local direction $\tau_i$ is determined as:

$$\tau_i = \frac{p_{i+1} - p_i}{||p_{i+1} - p_i||} \tag{5.4}$$

Then, two points $p_i'$ and $p_i''$ are defined as:

$$p_i \pm \frac{w_p}{2}(\tau_i \times n_i) \tag{5.5}$$

where $w_p$ is the width of the visualized path and $n_i$ is the local normal of the ground model at point $p_i$. The set made by the pairs $(p_i', p_i'')$ of all segment points defines the vertices of the corresponding quad strip as shown in Fig. 5.7.

### 5.3.2 Dynamic loading of point cloud clusters

A dynamic strategy is adopted for caching point cloud clusters in the VRAM of the graphic card to speed up rendering. Let $C$ be the total number of non-empty point cloud clusters (saved on disk) of $D^{N\times M}$, describing the static environment. An asynchronous and multithreaded algorithm keeps in the VRAM only the $L$ closest clusters to the current position of the user in the VR environment, while the number of rendered clusters at each frame is fixed to $V$, with

$$V \leq L \leq C \tag{5.6}$$

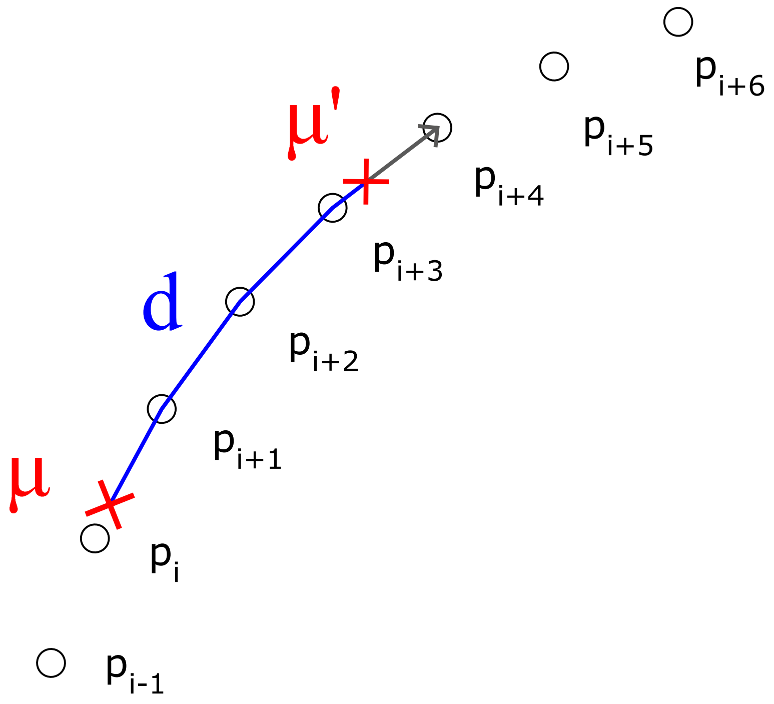


Figure 5.6: Path following example. White circles are the sampled points along the segments of the path. The sum of the blue lines connecting current position $\mu$ with the future position $\mu'$ is the distance traveled by the vehicle during the current frame.

The algorithm is divided in three phases. In the first phase, the $L$ clusters closest to the user are computed. Clusters already in memory that belong to the set of $L$-closest clusters are marked, while missing clusters are scheduled for loading. In the second phase, clusters in memory that do not belong to the set of $L$-closest clusters are marked as replaceable and associated to one scheduled cluster to be loaded in memory. In the third phase, a thread is created for each scheduled cluster to be loaded in memory. A thread-safe state vector is used to track the state of each cluster. As stated before, at each rendering loop only the $V$ closest cluster to the user are actually rendered.

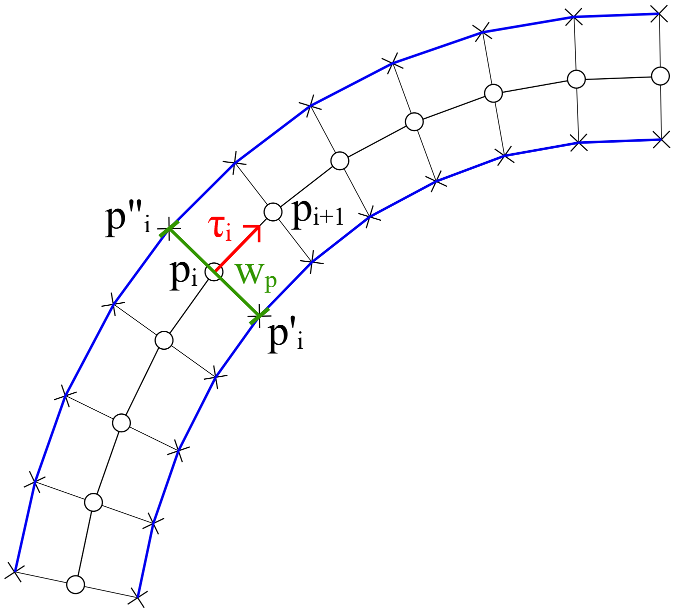


Figure 5.7: Path creation. White circles are the sampled points on the path, while blue crosses are the vertices of the associated quad strip used for rendering.

## 5.4 Collision detection with environment model

A number of efficient techniques are applied to achieve real-time collision detection of the vehicle 3D model with the large scale point cloud that represents the environment. The complete point cloud is filtered so that only points in the neighborhood of each vehicle path are considered for collision detection. In particular, only points $v^k{=}[v^k_x, v^k_y, v^k_z] \in P$ that satisfy all the following conditions

$$\begin{aligned} p_x^{min} - \delta \le v_x^k \le p_x^{max} + \delta \\ p_y^{min} - \delta \le v_y^k \le p_y^{max} + \delta \\ f_g(x,y) < v_z^k \le f_g(x,y) + \delta \end{aligned} \tag{5.7}$$

are considered in the collision test. Where $[p_x^{min}, p_y^{min}]$, $[p_x^{max}, p_y^{max}]$ are the minimum and maximum coordinates of all sampled points of the path, and $\delta$ is the diagonal of

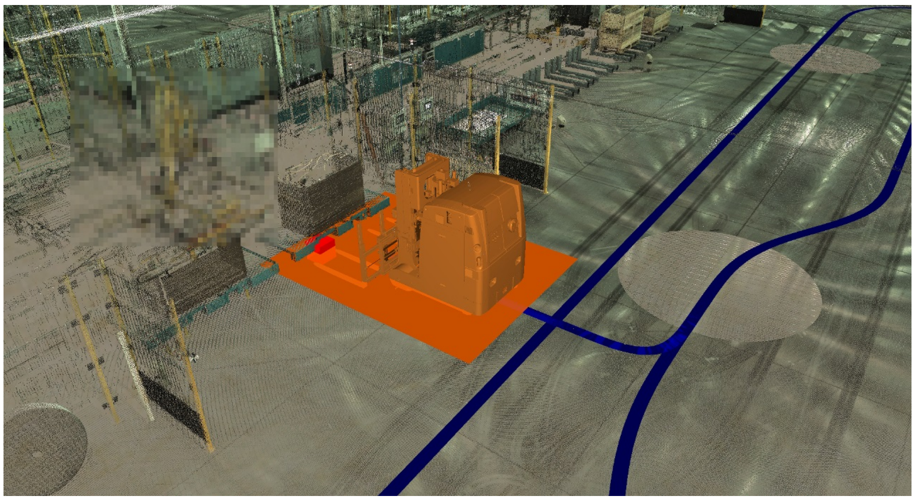


Figure 5.8: Collision with safety shapes. The collision point is highlighted with a red cube. Some regions of the image have been blurred to prevent copyright-protected content diffusion. Vehicle (dark orange). Safety zone (light orange).

the OBB of the volume obtained as the union between the vehicle OBB and the largest safety zone of the path. During animation points are further filtered by applying the same procedure described above to each segment of the path. Hence, each segment is associated to its own point cloud, that includes possible collision points, which are tested at runtime. The filtering procedure takes few seconds, and it is performed at the beginning of each animation path in a separate thread.

The collision detection algorithm works as follows. First, each point is tested to check if it collides with the safety shape of the vehicle in the current segment. If the test succeeds, the algorithm terminates reporting a collision with the safety shape. Otherwise, if the test fails, the OBB tree of the vehicle, described in section 5.2.1, is traversed starting from the root node. Each point is tested for collision with the OBB of the current node of the tree (test1). If collision occurs the procedure continues recursively in the subtrees of the current node. If a leaf node is reached and test1 reports a collision, a second collision test (test2) is performed between the point and the convex polyhedron that describes the geometry of the leaf node.

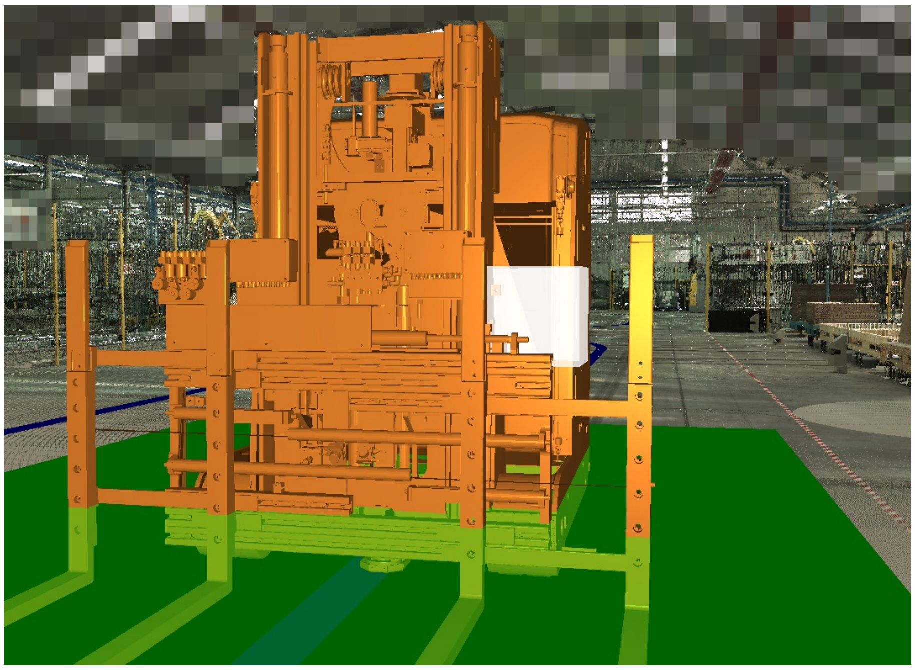


Figure 5.9: Collision with the OBB of a vehicle part. The collision point is highlighted with a white cube. Company plate and ceiling have been blurred to prevent copyright-protected content diffusion.

In summary, in the proposed scheme, three types of collision are reported: collisions with safety shapes (Fig. 5.8), collisions with the OBB of a vehicle part (Fig. 5.9), and collisions with a part of the convex decomposition of the vehicle (Fig. 5.10). Each type of collision is highlighted in a different way. Collisions with safety shapes are signaled by changing the color of the safety shape from green to light orange, and a red cube surrounding the collision point is displayed in red. Collisions with the OBB of a vehicle part are signaled by displaying a white cube surrounding the collision point (while the safety shapes are maintained in green color). Collisions with the convex decomposition of the vehicle are signaled by displaying a red cube surrounding the collision point (while the safety shapes are maintained in green color). Each collision event triggers the vehicle to stop until a resume command is issued by the user.

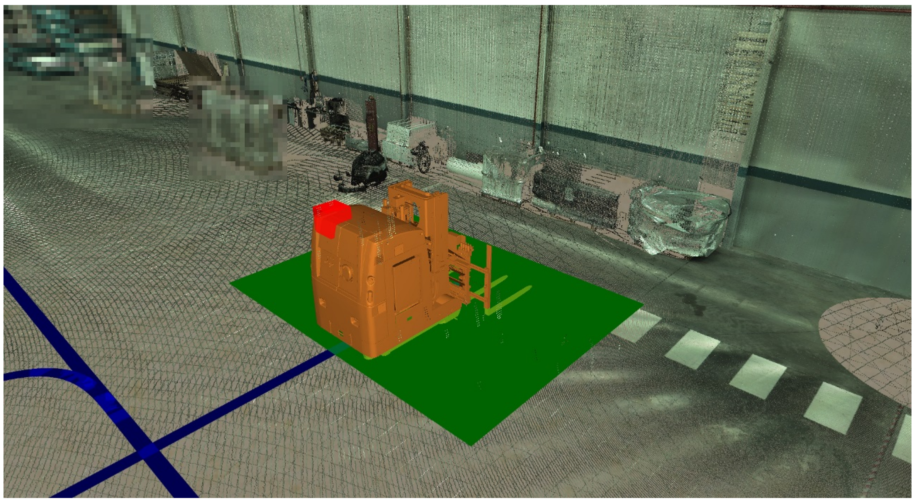


Figure 5.10: Collision with vehicle convex decomposition. The collision point is highlighted with a red cube. Ground goods have been blurred to prevent copyright-protected content diffusion.

## 5.5 Experiments

Data processing and visualization have been performed on a desktop Intel i7-5960x (3.00GHz) CPU with 64GB RAM, a Solid State Drive and a Nvidia Titan X 12GB graphics card. A first dataset consists of about 500 millions of points acquired from 63 scans. A second dataset consists of about 15 billions of points acquired from 1641 scans. The two datasets have been acquired from real warehouses where AGVs are connected to a centralized controller.

Thanks to the dynamic point cloud loading algorithm the performance of the VR system does not depend on the size of the input point cloud. Experiments were performed to compare the multithreaded system with a sequential algorithm. The sequential algorithm in each frame manages all clusters scheduled to be loaded in memory iteratively, without loading them asynchronously. The sequential algorithm causes a lower and unstable framerate and a less smooth user experience. Fig. 5.11 shows the framerates of both approaches running a scripted task, where the user

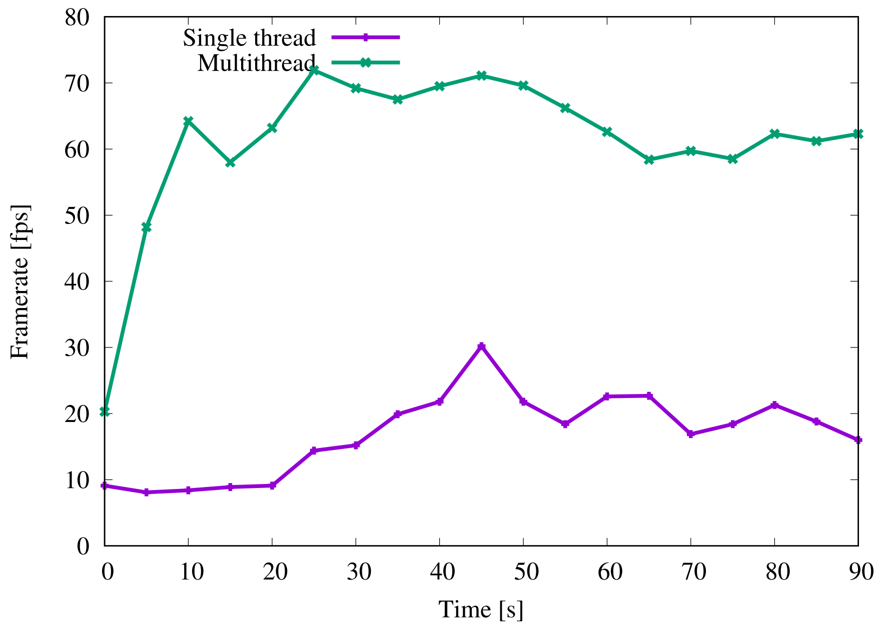


Figure 5.11: Comparison between the single and the multithread algorithm. The multithread algorithm achieves 60 fps during the scripted path, while the performance of the single thread algorithm is lower than 30 fps.

follows a predefined path, with five vehicles moving simultaneously. The average framerate of the parallel algorithm is over 60 fps, while the average framerate of the sequential algorithm is lower than 30 fps. Vehicles stops caused by a collision detection has been ignored during all the scripted tasks.

Vehicles used in the reported experiments are based on the tricycle kinematic model, with a front drive wheel and two passive back wheels. The chassis size is about 2 m x 1.5 m x 2 m, while the fork system is about 1.5 m, so that the total length of the vehicle is 3.5 m. The reference point of the vehicle for path following is located on the ground projection of the center of the passive wheels axis. The 3D model consists of 1 940 607 vertices, while the simplified model has 910 vertices. Figure 5.12 shows the experimental setup for the tests.

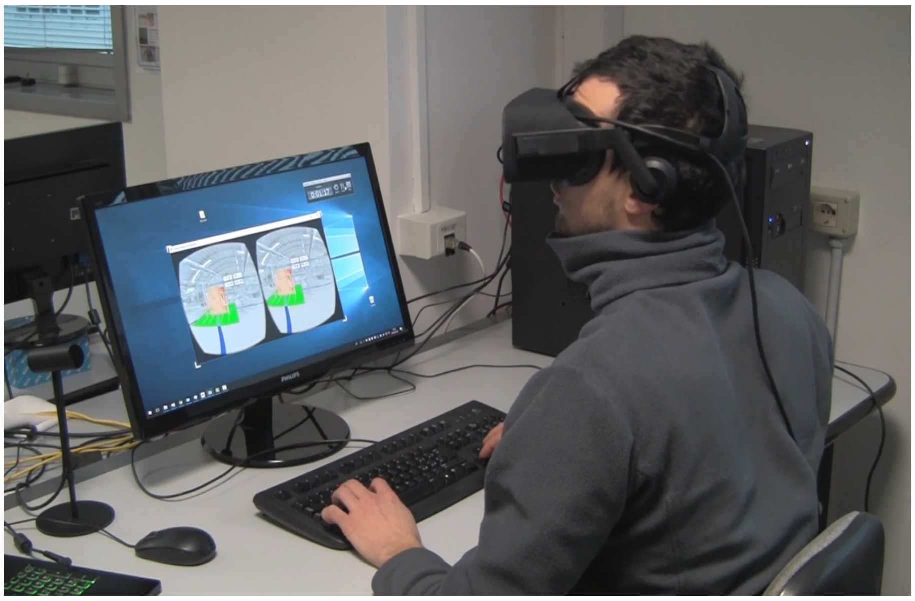


Figure 5.12: Experimental setup.

## 5.6 Discussion

A desktop VR system was presented for visualization of AGV and collision detection with large scale point clouds. Point clouds of industrial environments have been obtained from terrestrial laser scanning surveys. The system performs vehicle simulation by taking into account ground slope. AGV safety zones are also rendered for each segment of the path. A multithreaded approach is adopted to speed up rendering that exploits a caching technique of point cloud clusters. The VR environment supports real-time collision detection between the vehicle model and the point cloud, which is used for accessibility checking.

# Chapter 6

# Sensor-Based Optimization of TLS Setup on GPU

## 6.1 Introduction

A terrestrial laser scanning (TLS) survey is usually performed by technicians who determine the scan station positions from their experience without any guarantee of optimality. To improve the quality of the survey the scan station positions can be computed automatically by solving a constrained optimization problem. In this chapter a novel formulation of the optimal sensor placement problem is presented for TLS. Optimal sensor placement has been extensively investigated for visual sensors including directional and omnidirectional cameras [13, 21, 32, 9, 26, 79, 75, 80]. However, few works have investigated optimal sensor placement for TLS. With respect to past works the proposed approach includes a more elaborate sensor model. Indeed, the adopted sensor model simulates a number of fixed parameters having a strong influence on the laser measurement, as pointed out in [10, 37, 42, 64, 35, 65, 66], like the laser height from the ground, angular resolution, field of view, and sensor range. Moreover, the proposed approach includes a wider set of constraints with respect to past works. The two closest works are [2, 63]. The main difference is that in [2] and [63] constraints were imposed only on the edges of the map, while in this work optimization is performed on all cells of the ground. Furthermore, [2] only considers a constraint on the horizontal angle of incidence, and [63] does not consider the overlap constraint. In [36] a genetic algorithm was investigated in small scale environments, but coverage and overlap constraints were not included. Finally, the proposed approach exploits GPU acceleration, which enables experiments in large scale environments, with both external walls and internal structures. Other relevant works are [30, 34]. However, in [30] a method was presented that requires an initial scan of the environment, while in [34] a solution was presented for the specific problem of TLS next-best view planning for piping systems.

The mapping of the laser beams on the ground can be modeled by a gnomonic projection, where the radial scale increases more rapidly with the distance from the scanner than the transverse scale, i.e. there is a non uniform distribution of horizontal and vertical scan lines on the ground. Therefore, in the proposed approach a coverage constraint is imposed to guarantee a minimum density of horizontal scan lines (radial scale) on the ground. Indeed, a coverage estimation based on counting the number of points would be too much affected by the high density of points along the horizontal scan lines. An overlap constraint among the scans is also imposed to enable automatic point cloud alignment and registration, which is a standard post-processing step of a TLS survey when artificial targets are not used. Indeed, even though artificial targets ensure a very high registration quality, they have a huge impact on survey time. Moreover, not all scanners on the market support artificial targets. For this reasons, the overlap constraint imposes an order in the scan sequence, so that each scan can be registered with the union of all previous scans by applying standard algorithms for point cloud registration. Constraints also take into account occlusions and a maximum allowed incidence angles of the laser beams on the segments.

## 6.2 Problem formulation

This work aims to find the shortest sequence of scan stations that meets a coverage constraint and an overlap constraint given an input floor plan model, a desired minimum number **R** of horizontal scan lines per square meter on the ground, and an overlap threshold $\tau$. The floor plan is modeled as a set of line segments, representing walls and other elements higher than the scanner height, and it is encoded as a two dimensional $n \times m$ grid with cells of size $\delta$. The optimal solution is found by solving an optimization problem formulated as a modified version of the *Set Cover Problem*, which is NP-Complete [23]. Let $p_{i,j}$ be a binary value that indicates whether a scan has to be performed at position $(i, j)$, that is

$$p_{i,j} = \begin{cases} 1 & \text{if a scan station must be placed in cell } (i,j) \\ 0 & \text{otherwise} \end{cases}$$

Let also $\mathcal{S}$ be the set of all possible ordered finite scan sequences, having a length between 1 to $n \times m$, i.e.

$$\mathcal{S} \doteq \left\{ \begin{array}{c} s \doteq \{(i_1, j_1) \succ (i_2, j_2) \succ \ldots \succ (i_N, j_N)\} \ : \\ N \in \{1, \ldots, n{\times}m\} \ \wedge \ p_{i_t, j_t}{=}1 \ \forall t{=}1, \ldots, N \end{array} \right\}$$

where $(i, j){\succ}(i', j')$ indicates that scan at $(i, j)$ must be performed before scan at $(i', j')$. The optimal scan setup is computed as the shortest sequence of scan stations, i.e.

$$\underset{s \in \mathcal{S}}{\text{minimize}} \ |s|, \tag{6.1}$$

s.t.

$$\sum_{i,j=1}^{n,m} c_{[(k,l)|(i,j)]} \geq \mathbf{R}\delta^2, \quad \forall k, l \ : 1 \leq k \leq n, 1 \leq l \leq m \tag{6.2}$$

$$\frac{\sum\limits_{k,l=1}^{n,m} \min\left( \sum\limits_{v=1}^{t-1} c_{[(k,l)|(i_v, j_v)]}, c_{[(k,l)|(i_t, j_t)]} \right)}{\sum\limits_{k,l=1}^{n,m} c_{[(k,l)|(i_t, j_t)]}} \geq \tau, \ \forall t : 1 < t \leq N \tag{6.3}$$

Problem (6.1) finds the ordered scan sequence $(i_1, j_1) \succ \ldots \succ (i_N, j_N)$ with the minimum cardinality that satisfies a ground coverage constraint (6.2) and an overlap constraint (6.3). The coverage constraint (6.2) imposes a minimum number of horizontal scan lines per cell. Value $\hat{c}_{[(k,l)|(i,j)]}$ is defined as the number of horizontal scan lines in cell $(k, l)$ due to a scan station in cell $(i, j)$, and $c_{[(k,l)|(i,j)]}$ as the minimum between $\hat{c}_{[(k,l)|(i,j)]}$ and the upper bound $\mathbf{R}\delta^2$, i.e.

$$\begin{aligned}
&\hat{c}_{[(k,l)|(i,j)]}=(1-\Gamma_{i,j})p_{i,j}F_\delta(\delta(k-i),\delta(l-j))v_{[(k,l)|(i,j)]}\\
&c_{[(k,l)|(i,j)]}=\min(\hat{c}_{[(k,l)|(i,j)]},\mathbf{R}\delta^2)
\end{aligned} \tag{6.4}$$

where $F_\delta(x,y)$ is the coverage function defined in Section 6.2.1, and $v_{[(k,l)|(i,j)]}$ is a binary value that manages visibility, taking into account occlusion, sensor range, as well as horizontal and vertical angles of incidence (Section 6.2.2), i.e.

$$v_{[(k,l)|(i,j)]} = \begin{cases} 1 & \text{if } (k,l) \text{ is visible from } (i,j) \\ 0 & \text{otherwise} \end{cases} \tag{6.5}$$

Matrix $\Gamma_{i,j} \in \mathbb{R}^{n\times m}$ contains the rasterization of each input segment $\gamma$, with $\Gamma_{i,j} = 1$ if a segment exists in cell $(i,j)$ ($\Gamma_{i,j} = 0$ otherwise). Factor $(1-\Gamma_{i,j})$ in (6.4) is used to ensure that a scan station is not placed in a cell occupied by a segment. The overlap constraint (6.3) is expressed as a function of the cell coverage $c_{[(k,l)|(i,j)]}$ as well. In particular, the overlap constraint is satisfied if, for each scan $t$, the relative value of ground coverage in common between scan $t$ and all previous scans is higher than the threshold value $\tau$. Indeed, the numerator of (6.3) contains the sum across all the cells of the minimum between the scan coverage in a cell, due to scan $t$, and the sum of the coverage values in the same cell, due to all previous scans.

### 6.2.1 Laser model and ground coverage function

In order to properly simulate the scanner, the following parameters are considered: laser height from the ground $\mathbf{h}$ (Fig. 6.4, right), horizontal and vertical angular resolution $\mathbf{r}$, maximum allowed beam incidence angle on surfaces $\theta_{max}$, minimum and maximum ranges ($\mathbf{d}^-$,$\mathbf{d}^+$) and field of view. The ground coverage function $F_\delta(x,y)$ of a scan station in cell $(0,0)$ estimates the number of horizontal scan lines in a cell of size $\delta$, centered in $(x,y)$, as the ratio between the difference of the vertical angles of the outer beams that hit the cell, and the vertical angular resolution, i.e.

$$F_\delta(x,y) = \frac{\tan^{-1}(\frac{2\sqrt{x^2+y^2}+\delta}{2\mathbf{h}}) - \tan^{-1}(\frac{2\sqrt{x^2+y^2}-\delta}{2\mathbf{h}})}{\mathbf{r}} \tag{6.6}$$

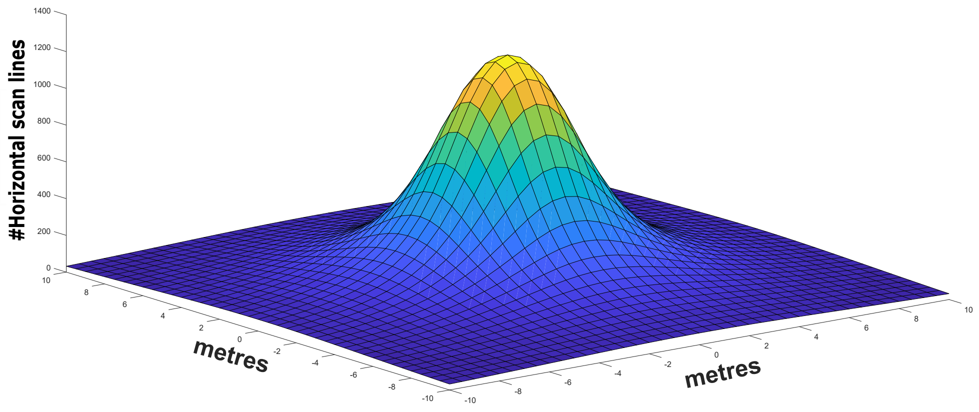


Figure 6.1: Ground Coverage $F_\delta(x, y)$ with $\mathbf{r}$ = 1.25 mrad and $\mathbf{h}$ = 2 m.

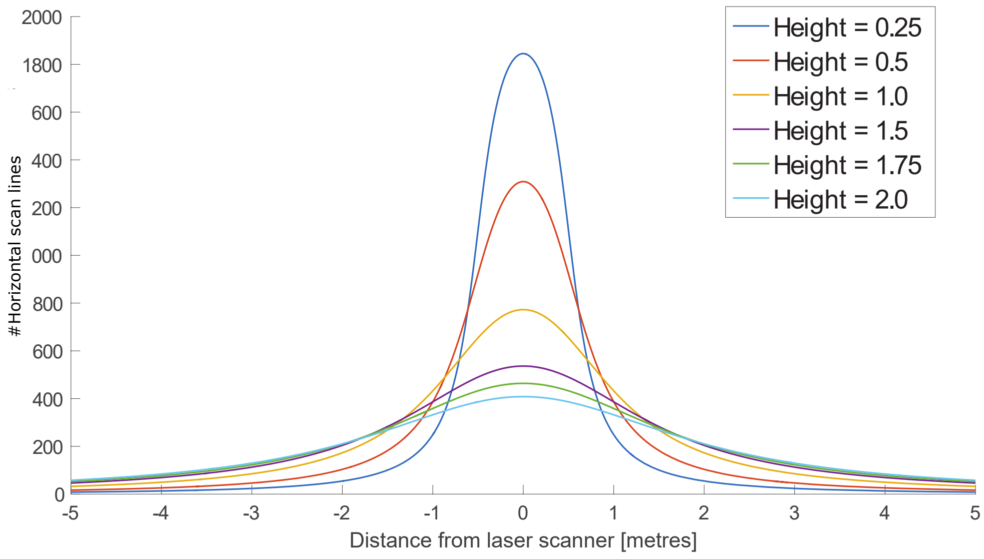


Figure 6.2: Ground Coverage with respect to the horizontal distance from the laser origin, for multiple values of height with $\mathbf{r}$ = 1 mrad.

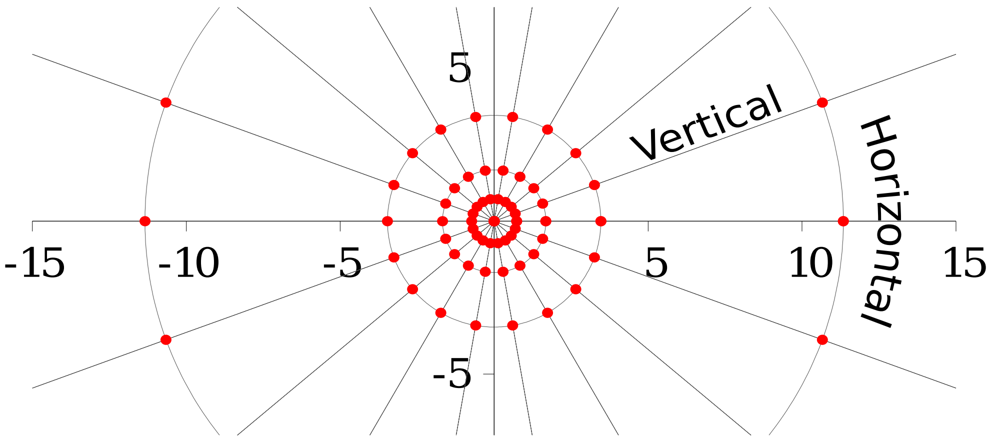


Figure 6.3: Example of mapping of the laser beams on the ground with resulting horizontal and vertical scan lines.

Fig. 6.1 shows the coverage function and the high dependency on the horizontal distance from the laser origin.

It can be shown that **h** does not affect the total amount of horizontal scan lines on the ground, as the integral of $F_\delta(x, y)$ over the whole plane $\mathbb{R}^2_{x,y}$ does not depend on **h**, i.e. the area under all curves in Fig. 6.2 is constant. However, **h** has a strong influence on the shape of the ground coverage function, as a small change in height causes a significant change in the distribution of the laser beams on the ground, thus emphasizing the importance of an accurate laser model.

### 6.2.2 Visibility

Visibility takes into account occlusions by segments, sensor range, as well as angles of incidence of the laser beams on ground and segments. The occlusion by each floor plan segment $\gamma = (P_h, P_{h+1})$, where $P_h$ and $P_{h+1}$ are the two 2D vertices of the segment, is formulated as follows. Let $P_O$ be the vertical projection of the scanner origin on the ground. Let also $\overline{P_h P_{h+1}}$ be the line through $P_h$ and $P_{h+1}$. Similarly, let $\overline{P_O P_h}$ and $\overline{P_O P_{h+1}}$ be the two lines passing through the other pairs of points defined above. The occluded region of the ground due to segment $\gamma$ is computed as the intersection

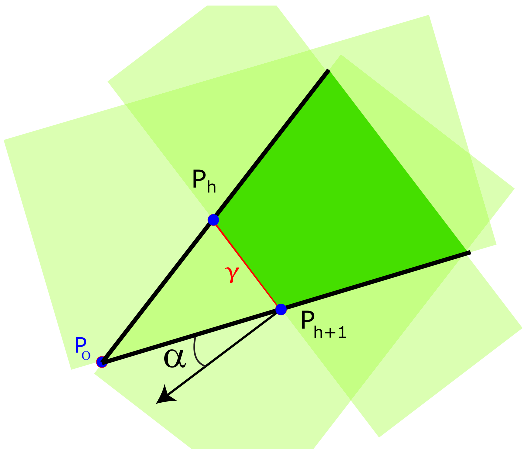


Figure 6.4: Left: occluded area on the ground (dark green) due to segment $\gamma$ ($\alpha$ is the horizontal angle of incidence) obtained as the intersection of three half planes (light green). Right: Vertical angle of incidence $\beta$ and laser height **h**.

between three half planes, as shown in Fig. 6.4 (left), aligned points are handled as a special case. The first half plane is the one that does not contain $P_O$ with respect to $\overline{P_h P_{h+1}}$, the second is the one that contains $P_{h+1}$ with respect to $\overline{P_O P_h}$, the third is the one that contains $P_h$ with respect to $\overline{P_O P_{h+1}}$. Let $W^{\gamma}_{[(k,l)|(i,j)]}$ be a function which yields 0 if cell $(k,l)$ is occluded by segment $\gamma$ from a scan station located in $(i,j)$, and 1 otherwise.

Then, a binary output function $W_{[(k,l)|(i,j)]}$ can be defined, which yields 0 if cell $(k,l)$ is occluded by at least one segment from a scan station located in $(i,j)$, i.e.

$$W_{[(k,l)|(i,j)]} = \prod_{\{\gamma\}} W^{\gamma}_{[(k,l)|(i,j)]} \tag{6.7}$$

Visibility also includes a condition on the angle of incidence of the laser beam on the segments and on the ground. Let $\alpha_{[(k,l)|(i,j)]}$ be the horizontal angle of incidence (Fig. 6.4, left), i.e. the angle between the normal to the segment in cell $(k,l)$ and the horizontal component of the laser beam from a scan station in cell $(i,j)$. Then, a binary output function $A^{H}_{[(k,l)|(i,j)]}$ can be defined, that yields 0 if the horizontal angle

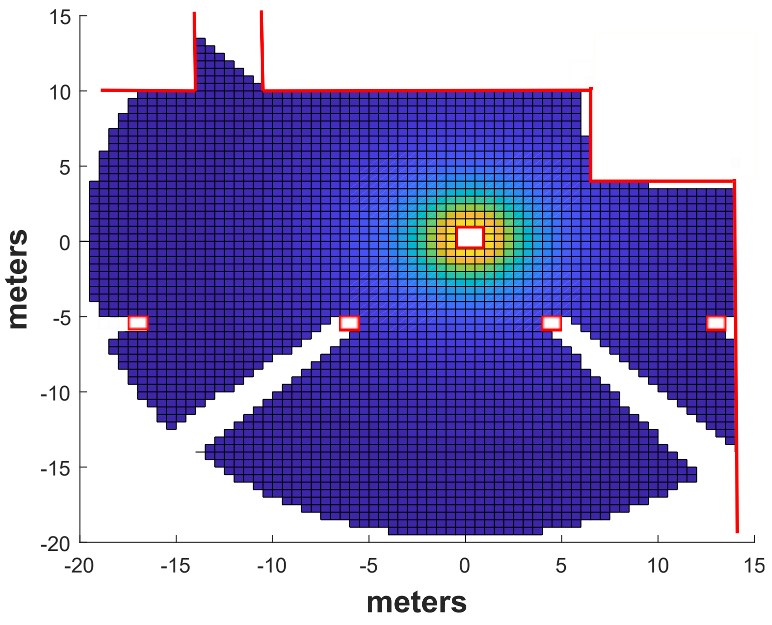


Figure 6.5: Example scan coverage $F_\delta(x, y)$ with visibility (top view). Segments are in red, $\mathbf{r}$ = 1.25 mrad, $\mathbf{h}$ = 2 m, $\theta_{max}$ = 84.5°, $\theta_{min}$ = 20°.

of incidence on a segment is greater than threshold $\theta_{max}$, i.e.

$$A^H_{[(k,l)|(i,j)]} = \begin{cases} 0 \text{ if } \Gamma_{k,l}=1 \wedge \alpha_{[(k,l)|(i,j)]} > \theta_{max} \\ 1 \text{ otherwise} \end{cases} \tag{6.8}$$

Similarly, the vertical angle of incidence $\beta_{[(k,l)|(i,j)]}$ (Fig. 6.4, right) is defined as the angle between the normal to the ground in cell $(k,l)$ and the laser beam from a scan station in $(i, j)$. Then, a binary output function $A^V_{[(k,l)|(i,j)]}$ can be defined to model the field of view, that yields 1 if $\beta_{[(k,l)|(i,j)]}$ is lower than $\theta_{max}$ and greater than a threshold $\theta_{min}$, i.e.

$$A^V_{[(k,l)|(i,j)]} = \begin{cases} 1 \text{ if } \theta_{min} < \beta_{[(k,l)|(i,j)]} < \theta_{max} \\ 0 \text{ otherwise} \end{cases} \tag{6.9}$$

Visibility considers the sensor range as well, by introducing a binary output function

$D_{[(k,l)|(i,j)]}$, which yields 1 if the measured point is within the sensor range, i.e.

$$D_{[(k,l)|(i,j)]} = \begin{cases} 1 & \text{if } \mathbf{d}^- < \sqrt{\delta^2(i-k)^2 + \delta^2(j-l)^2 + \mathbf{h}^2} < \mathbf{d}^+ \\ 0 & \text{otherwise} \end{cases} \tag{6.10}$$

Finally, (6.7),(6.8),(6.9) and (6.10) are combined together to define visibility (6.5), i.e.

$$v_{[(k,l)|(i,j)]} \doteq W_{[(k,l)|(i,j)]} A^H_{[(k,l)|(i,j)]} A^V_{[(k,l)|(i,j)]} D_{[(k,l)|(i,j)]} \tag{6.11}$$

meaning that a cell $(k,l)$ on the ground is visible from a scan station in cell $(i,j)$ if it is not occluded by any segment, if the horizontal and vertical angles of incidence are within their limits, and if the cell is within the laser range. An example is shown in Fig. 6.5.

## 6.3 Parallel implementation on GPU

Like in previous works, the optimization problem (6.1) was solved using a standard greedy iterative algorithm. The algorithm operates on a 2D cumulative coverage grid, where each value $g_{t,(k,l)}$ contains the total cumulative number of horizontal scan lines in cell $(k,l)$ up to iteration $t$. Cells are initialized with zeroes at $t$=1. The maximum value for a cell is limited to the objective coverage $\mathbf{R}\delta^2$. A set of 2D floor plan segments is given as input to the algorithm. Visibility map values $v_{[(k,l)|(i,j)]}$ are pre-computed in parallel on GPU. At each iteration, the algorithm selects the scan station position which maximizes the total coverage gain. The procedure is repeated until all cells have a value equal to the objective coverage, or no further gain can be achieved. In particular, at iteration $t$, for each cell $(k,l)$ scanned from cell $(i,j)$, a cell coverage gain $I_{t,[(k,l)|(i,j)]}$ and a cell overlap contribution $O_{t,[(k,l)|(i,j)]}$ are computed as:

$$\begin{aligned} I_{t,[(k,l)|(i,j)]} &= \min\Big(c_{[(k,l)|(i,j)]}, \mathbf{R}\delta^2 - g_{t,(k,l)}\Big) \\ O_{t,[(k,l)|(i,j)]} &= \min\big(c_{[(k,l)|(i,j)]}, g_{t,(k,l)}\big) \end{aligned} \tag{6.12}$$

Hence, the total coverage gain $\mathcal{I}_{t,(i,j)}$ and the total overlap $\mathcal{O}_{t,(i,j)}$, at iteration $t$, for scan position $(i, j)$ are computed as:

$$\begin{aligned} \mathcal{I}_{t,(i,j)} &= \sum_{k,l} I_{t,[(k,l)|(i,j)]} \\ \mathcal{O}_{t,(i,j)} &= \frac{\sum_{k,l} O_{t,[(k,l)|(i,j)]}}{\sum_{k,l} c_{[(k,l)|(i,j)]}} \end{aligned} \tag{6.13}$$

Equation (6.12) is computed on GPU in parallel for each cell $(k,l)$. Parallel reductions are then used to compute (6.13). The scan position at iteration $t$ is then selected in $(i_t, j_t)$, so that

$$(i_t, j_t) = \underset{\{(i,j)\,|\,\mathcal{O}_{t,(i,j)}>\tau\}}{\operatorname{argmax}} \mathcal{I}_{t,(i,j)} \tag{6.14}$$

with $p_{i_t,j_t} = 1$. Finally, the coverage grid is updated for next iteration $t+1$ as

$$g_{t+1,(k,l)} = g_{t,(k,l)} + I_{t,[(k,l)|(i_t,j_t)]} \tag{6.15}$$

The solution of (6.14) may not be unique as GPU scheduling is non-deterministic. Therefore, the execution on the same input data may lead to different optimal scan stations sequences, still with similar cardinality. The algorithm was implemented in CUDA/C++ and run on an Intel Core i7-5960X CPU @3.00 GHz, with 64 GB RAM, and an NVIDIA GeForce GTX TITAN X GPU, with 12 GB RAM.

## 6.4 Experimental evaluation

Experiments were carried out in three scenarios (two large scale environments, and one smaller scene), obtained from real floor plans of industrial warehouses. Ground size was about67500 $\mathrm{m}^2$ in scenario 1,25120 $\mathrm{m}^2$ in scenario 2, and600 $\mathrm{m}^2$ in scenario 3. Scanner parameters were as follows: **h**=2 m, **r**=1.25 mrad, $\mathbf{d}^-$=0.4 m, $\mathbf{d}^+$=120 m, $\theta_{max}$=87.5°, $\theta_{min}$=0°. The overlap constraint was set to $\tau$=33% in each experiment.

Figures 6.6 and 6.11 show the scan station positions computed by the proposed approach in scenario 1 and scenario 2. The proposed method achieved the required coverage with a practicable number of scan positions. Some scan stations are close to each other or near walls, as in the final iterations the greedy algorithm selects scan

Table 6.1: Number of scan positions and computation time ($\mathbf{R} = 100$)

| Scenario | $\delta$ (m) | Scan no. | GPU Time (min) | CPU Time (min) |
|---|---|---|---|---|
| 1 | 0.50 | 369 | 99.8 | > 1 day |
| | 0.75 | 355 | 19.4 | 931.6 |
| | 1.00 | 359 | 7.0 | 290.3 |
| | 1.50 | 353 | 1.9 | 59.2 |
| 2 | 0.50 | 162 | 12.8 | > 1 day |
| | 0.75 | 168 | 2.8 | 100.2 |
| | 1.00 | 160 | 0.9 | 30.2 |
| | 1.50 | 155 | 0.3 | 6.9 |

Table 6.2: Number of scan positions and computation time ($\delta$=0.5 m)

| Scenario | **R** | Scan no. | GPU Time (min) |
|---|---|---|---|
| 1 | 10 | 138 | 42.1 |
| | 25 | 241 | 71.5 |
| | 50 | 367 | 101.0 |
| | 75 | 481 | 125.1 |
| | 100 | 582 | 149.0 |
| 2 | 10 | 65 | 5.3 |
| | 25 | 108 | 8.5 |
| | 50 | 163 | 12.5 |
| | 75 | 214 | 16.7 |
| | 100 | 255 | 20.1 |

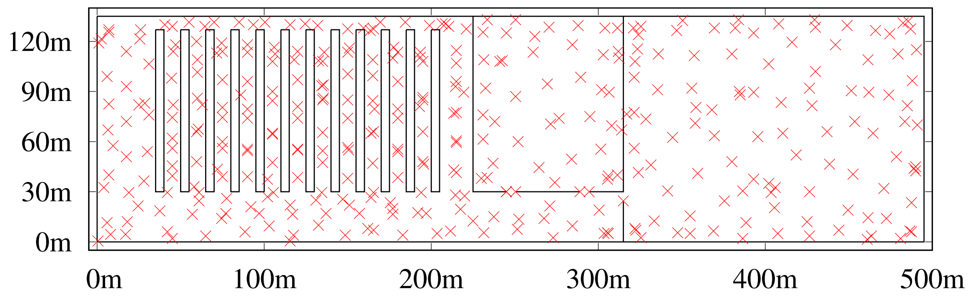


Figure 6.6: Optimized scan station positions in scenario 1, with $\delta = 0.5$ m and $\mathbf{R} = 50.0$. A total of 367 scan positions were computed (marked by crosses).

stations to observe small regions that are still uncovered. It can be observed that the distance to walls is always at least one cell size (0.5 m), which allows enough space for scanner placing. Table 6.1 reports results at different values of cell size $\delta$, with fixed **R**. A lower cell size allows scan stations to be placed with less uncertainty. However, computation time increases as cell size diminishes, as displayed in Fig. 6.7. Moreover, the number of scan positions is almost unaffected by cell size, as expected. Also, a comparison of the performance of the parallel algorithm (GPU) was carried out against a sequential algorithm on CPU. It can be deduced, from Table 6.1, that the parallel algorithm achieves a speed-up of at least 20. Table 6.2 shows results at different **R** values, with fixed cell size $\delta$=0.5 m. Both computation time and number of scan stations increase with the objective coverage **R**. The increase in the number of scan stations is linear, with coefficients of determination around 0.99 in both cases, as shown in Fig. 6.8. A linear increase is expected, because (without occlusions) each scan station increases the coverage by the same amount. To highlight the effectiveness of this method over a manual approach, two experienced technicians has been asked to perform a manual placement of the scan stations on a map of the floor plan of scenario 2. Technicians were asked to achieve a coverage of at least one point every 2 cm, which corresponds to an objective coverage of about **R**=50. The first technician (Fig. 6.9) placed only 83 scan station positions. He obtained the required coverage

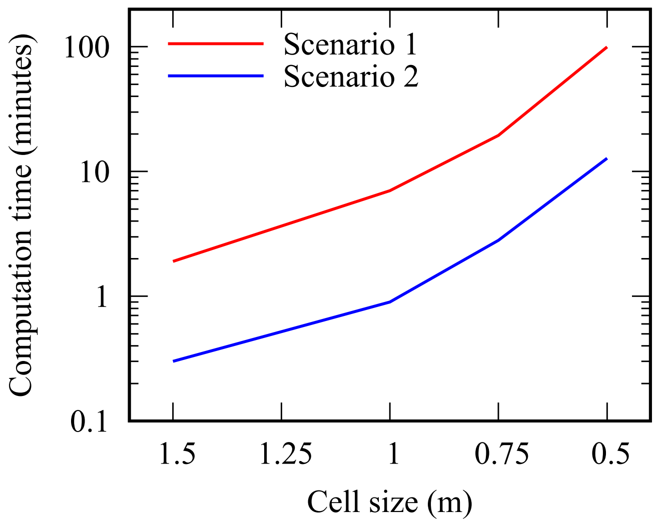


Figure 6.7: Computation time with respect to cell size $\delta$, with $\mathbf{R} = 100$ (logarithmic scale).

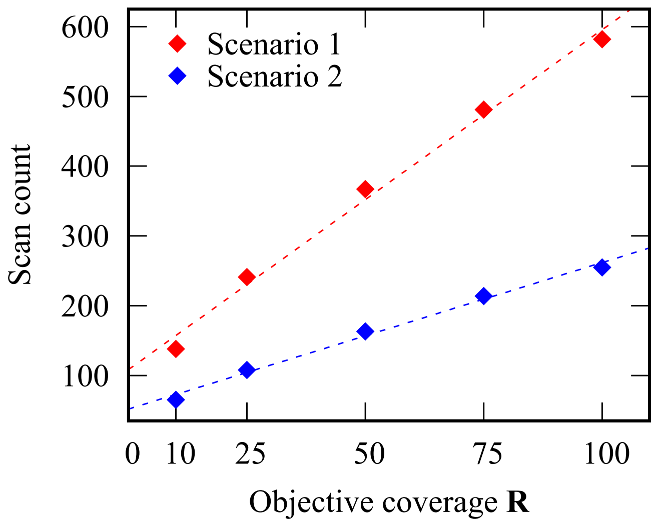


Figure 6.8: Number of scan stations with respect to **R** and linear regressions, with $\delta = 0.5$.

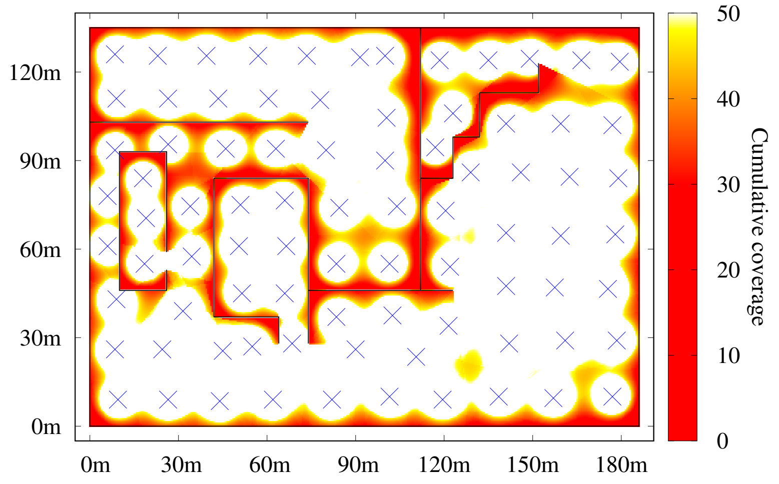


Figure 6.9: $g_{t,(k,l)}$ for manual scan placement (technician 1, 83 total positions). Color scale goes from white (fully covered cells) to red (uncovered cells).

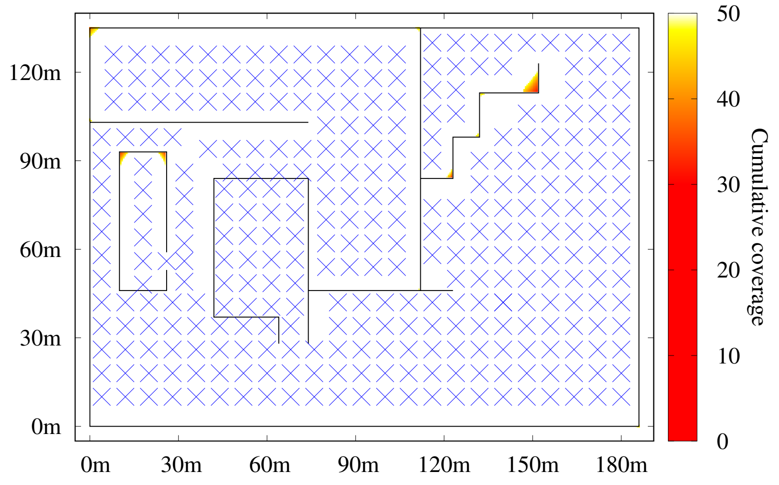


Figure 6.10: $g_{t,(k,l)}$ for manual scan placement (technician 2, 318 total positions).

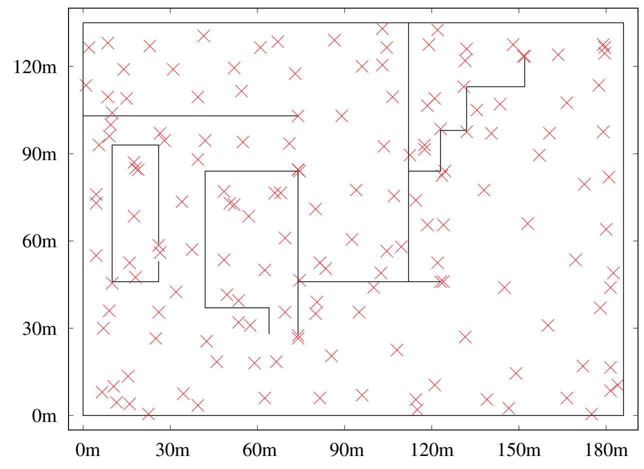


Figure 6.11: Optimized scan station positions in scenario 2, with $\delta = 0.5$ m and $\mathbf{R} = 50.0$. A total of 163 scan positions were computed (marked by crosses).

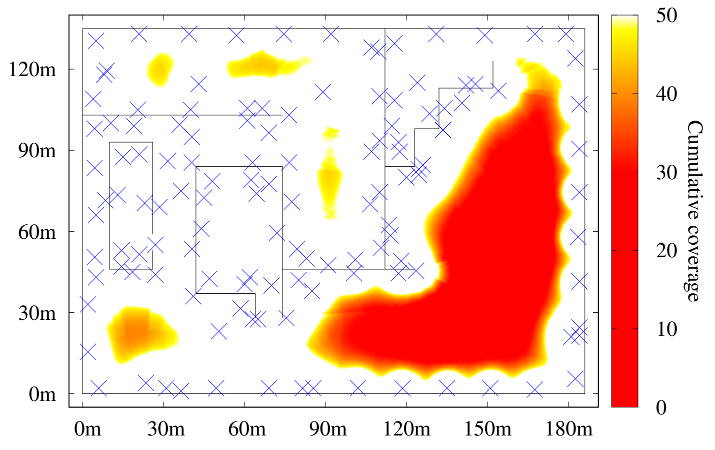


Figure 6.12: Optimized scan station positions in scenario 2 ($\delta = 0.5$ m, $\mathbf{R} = 50.0$), with coverage and overlap constraints limited to floor plan segments [2, 63]. A total of 130 scan positions were computed (marked by crosses).

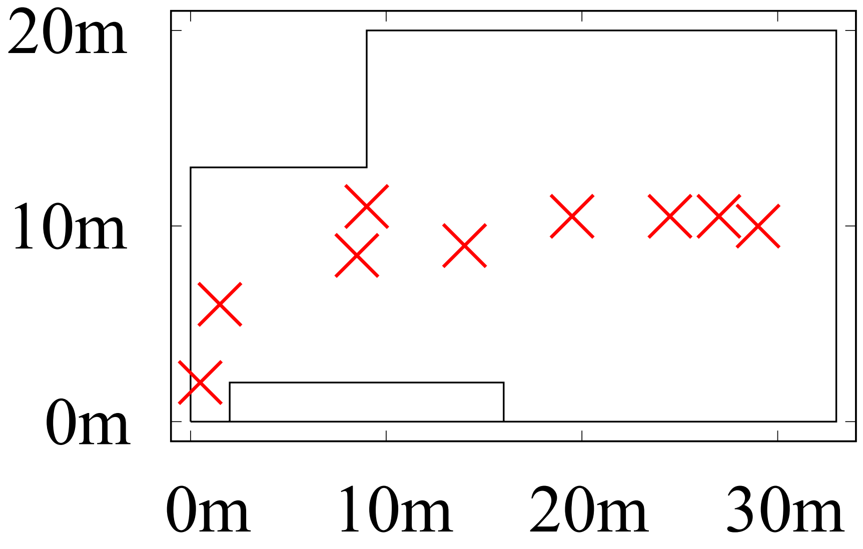


Figure 6.13: Optimized scan stations in scenario 3 (9 total positions).

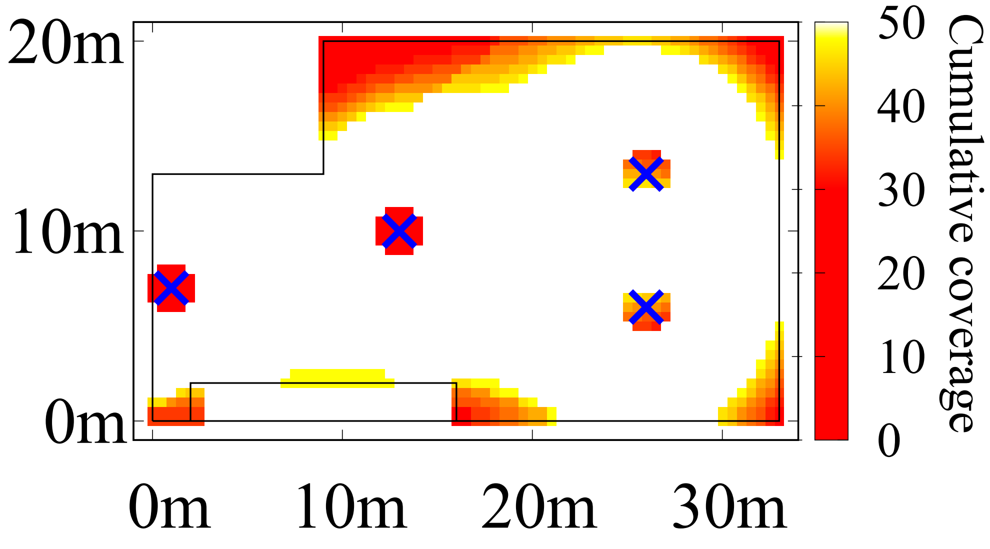


Figure 6.14: Cumulative coverage $g_{t,(k,l)}$ from manual scan placement (technician 1, 4 total positions).

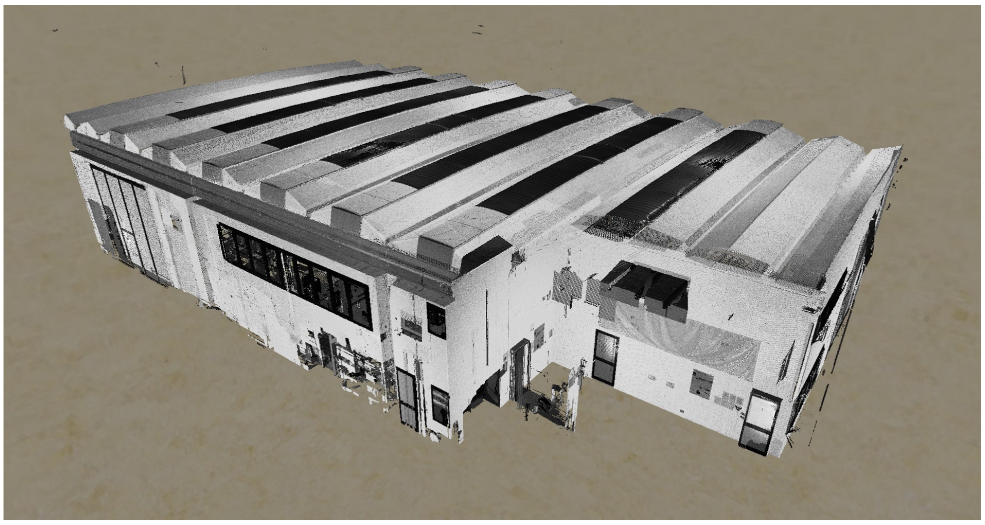


Figure 6.15: Point cloud from survey performed with Leica Scanstation P30 using the optimized scan stations ($\theta_{min}$=35°).

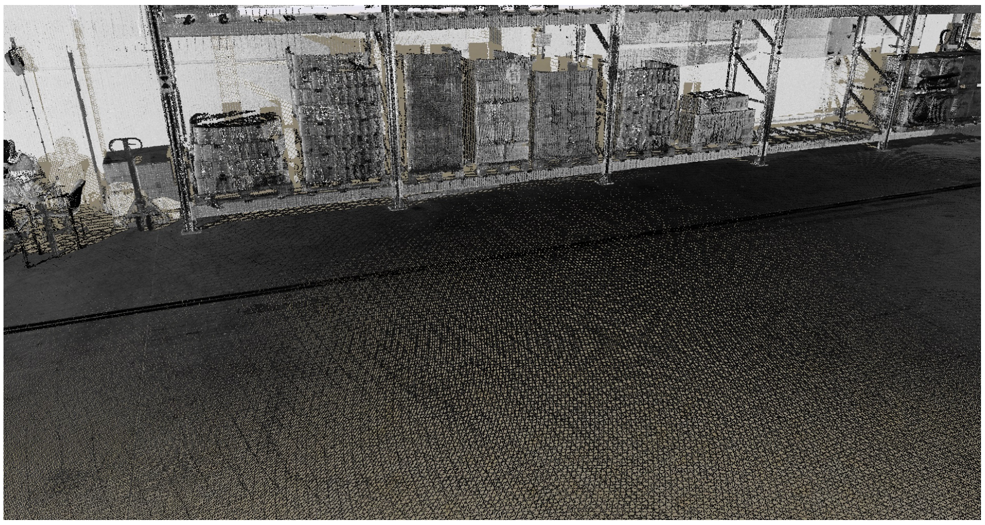


Figure 6.16: Detail of ground coverage from survey performed with Leica Scanstation P30 using the optimized scan stations ($\theta_{min}$=35°).

in most open areas, but he failed to reach it near walls and corners. The second technician (Fig. 6.10) was more conservative. He obtained the required coverage almost everywhere by placing the scan stations on a thick regular grid. However, he placed 318 scan stations, almost twice the number of the proposed method. A comparison with previous approaches was also carried out. As shown in Fig. 6.12, if optimization is performed only on the floor plan segments [2, 63], some parts of the ground are not properly covered. Another experiment was executed, in scenario 3, to show the results of a real TLS survey performed with a Leica Scanstation P30 using the optimized scan station positions. The required coverage is achieved everywhere (Fig. 6.13), whereas a manual placement of the scan stations leads to a sub-optimal coverage (Fig. 6.14). Figure 6.15 shows an image of the registered point cloud obtained from the survey, while Figure 6.16 shows a detail of the ground coverage.

## 6.5 Discussion

A novel formulation of the set cover problem was presented to find the optimal placement of scan stations in a TLS survey. As practical application, technicians can perform the survey following the ordered sequence of scan stations. The approach exploits GPU acceleration. The constraints of the optimization algorithm guarantee a minimum coverage of the ground and a minimum overlap among scans. A comparison against a manual placement performed by technicians was carried out to highlight the effectiveness of the method.

# Chapter 7

# Conclusion

This dissertation presented a novel workflow for the design of automated warehouses that combines traditional automation deployment with modern 3D technologies. Novel algorithms for ground segmentation, floorplan generation and real-time AGV collision detection in large scale point clouds have been presented as fundamental steps of the proposed workflow. Furthermore, a novel approach for the optimization of TLS measurement setup has been proposed, that includes realistic sensor-based constraints like coverage and overlap.

Experiments have been performed on datasets acquired in real-case scenarios. Results have been actively used to speed-up plant design, showing disruptive benefits.

As an example, it has been estimated that the adoption of the proposed workflow in the largest dataset (1641 scans, ~15 billions of points) saved about six months of work of two technicians, thus having huge impact on the installation cost. Furthermore, the impact of the installation on the customer production activities has been negligible. In fact, the Autonomous Guided Vehicles paths with all the operation points (~70.000 pallet positions) and safety areas have been designed and validated before the installation. Transition from human-operated forklifts to Autonomous Guided Vehicles has been drastically simplified, as it required neither to stop warehouse activities, nor to empty racks as in the past. Moreover, similar results have been obtained in several other warehouses, thus confirming the validity of the proposed approach.

Hence, the proposed workflow has become a standard in Elettric80 for the design and development of automated warehouses. Future work will investigate approaches for the automatic extraction of AGV operation points to further improve the proposed workflow. The 3D visualization tool described in Chapter 5 will also be integrated in the traffic management software, to show a real-time 3D representation of automated warehouses. Novel techniques for efficient visualization of large scale point clouds (billions of points) will be investigated. Level of Detail techniques will be exploited for simultaneous visualization of complex vehicle models.

New state of the art sensors that exploit visual odometry to speed-up point cloud alignment, like the Leica RTC 360, will also be considered to perform the survey, as they significantly reduce the amount of time spent for the point cloud registration.

# Bibliography

[1] A. Adan and D. Huber. «3D Reconstruction of Interior Wall Surfaces under Occlusion and Clutter». In: *International Conference on 3D Imaging, Modeling, Processing, Visualization and Transmission*. 2011, pp. 275–281.

[2] J. Ahn and K. Wohn. «Interactive scan planning for heritage recording». In: *Multimedia Tools and Applications* 75.7 (Apr. 2016), pp. 3655–3675.

[3] R. Ambrus, S. Claici, and A. Wendt. «Automatic Room Segmentation From Unstructured 3-D Data of Indoor Environments». In: *IEEE Robotics and Automation Letters* 2.2 (2017), pp. 749–756.

[4] K. Babacan et al. «Towards object driven floor plan extraction from laser point cloud». In: *ISPRS - Intl Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences* XLI-B3 (2016).

[5] P. Beinschob et al. «Semi-automated map creation for fast deployment of AGV fleets in modern logistics». In: *Robotics and Autonomous Systems* 87 (2017), pp. 281 –295.

[6] Y. Belkhouche, P. Duraisamy, and B. Buckles. «Ground Extraction from Terrestrial LiDAR Scans Using 2D-3D Neighborhood Graphs». In: *Advances in Visual Computing: 11th International Symposium, ISVC, Las Vegas, NV, USA, Part II*. 2015, pp. 655–663.

[7] J. Berglund et al. «Using 3D laser scanning to support discrete event simulation of production systems: lessons learned». In: *Proceedings of the Winter Simulation Conference*. 2014, pp. 2990–2999.

[8] J. M. Biosca and J. L. Lerma. «Unsupervised robust planar segmentation of terrestrial laser scanner point clouds based on fuzzy clustering methods». In: *ISPRS Journal of Photogrammetry and Remote Sensing* 63.1 (2008), pp. 84–98.

[9] T. Bodor et al. «Optimal camera placement for automated surveillance tasks». In: *Journal of Intelligent and Robotic System*. Vol. 50. Nov. 2007, pp. 257–295.

[10] W. Boehler, V. Bordas, and A. Marbs. «Investigating Laser Scanner Accuracy». In: *XIXth CIPA Symposium*. Sept. 2003.

[11] Dorit Borrmann et al. «A mobile robot based system for fully automated thermal 3D mapping». In: *Advanced Engineering Informatics* 28 (July 2014).

[12] G. Bruder, F. Steinicke, and A. Nüchter. «Poster: Immersive point cloud virtual environments». In: *IEEE Symposium on 3D User Interfaces (3DUI)*. 2014, pp. 161–162.

[13] K. Chakrabarty et al. «Grid coverage of surveillance and target location in distributed sensor networks». In: *IEEE Transaction on Computers*. Vol. 51. Dec. 2002, pp. 1448–1453.

[14] T. Chen et al. «3D LIDAR-based ground segmentation». In: *The First Asian Conference on Pattern Recognition*. 2011, pp. 446–450.

[15] T. Chen et al. «Gaussian-Process-Based Real-Time Ground Segmentation for Autonomous Land Vehicles». In: *Journal of Intelligent & Robotic Systems* 76.3 (2014), pp. 563–582.

[16] T. Chen et al. «Sparse Gaussian process regression based ground segmentation for autonomous land vehicles». In: *The 27th Chinese Control and Decision Conference (CCDC)*. 2015, pp. 3993–3998.

[17] M. C. Codrea and O. S. Nevalainen. «Note: An algorithm for contour-based region filling». In: *Computers & Graphics* 29.3 (2005), pp. 441 –450.

[18] O. Conrad et al. «System for Automated Geoscientific Analyses (SAGA) v. 2.1.4». In: *Geosci. Model Dev.* 8 (2015), pp. 1991–2007.

[19] B. Douillard et al. «On the segmentation of 3D LIDAR point clouds». In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2011, pp. 2798–2805.

[20] K. Eilers and J. Rossmann. «Modeling an AGV based facility logistics system to measure and visualize performance availability in a VR environment». In: *Proceedings of the Winter Simulation Conference*. 2014, pp. 367–375.

[21] U. Erdem and S. Sclaroff. «Automated camera layout to satisfy task-specific and floor plan-specific coverage requirements». In: *Computer Vision and Image Understanding*. Vol. 103. Sept. 2006, pp. 156–169.

[22] D. Eriksson and E. Shellshear. «Approximate distance queries for path-planning in massive point clouds». In: *11th International Conference on Informatics in Control, Automation and Robotics (ICINCO)*. Vol. 2. 2014, pp. 20–28.

[23] R. Gandhi, S. Khuller, and A. Srinivasa. «Approximation algorithms for partial covering problems». In: *J. Algorithms*. Vol. 53. 2004, pp. 55–84.

[24] M. Giorgini and J. Aleotti. «Visualization of AGV in Virtual Reality and Collision Detection with Large Scale Point Clouds». In: *IEEE 16th International Conference on Industrial Informatics*. July 2018, pp. 905–910.

[25] M. Giorgini, F. Barbieri, and J. Aleotti. «Ground Segmentation From Large-Scale Terrestrial Laser Scanner Data of Industrial Environments». In: *IEEE Robotics and Automation Letters* 2.4 (2017), pp. 1948–1955.

[26] J.J. Gonzalez-Barbosa et al. «Optimal camera placement for total coverage». In: *IEEE Intl Conference on Robotics and Automation*. Kobe, Japan, May 2009, pp. 844–848.

[27] P. Gordon and T. Charles. «Terrestrial Laser Scanners». In: (Jan. 2009).

[28] M. Gregor, J. Herčko, and P. Grznár. «The factory of the future production system research». In: *21st International Conference on Automation and Computing (ICAC)*. 2015, pp. 1–6.

[29] A. Harakeh, D. Asmar, and E. Shammas. «Ground segmentation and occupancy grid generation using probability fields». In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2015, pp. 695–702.

[30] M. Heidari Mozaffar and M. Varshosaz. In: *The Photogrammetric Record* 31.156 (Dec. 2016), pp. 374 –393.

[31] M. Himmelsbach, F. V. Hundelshausen, and H. J. Wuensche. «Fast segmentation of 3D point clouds for ground vehicles». In: *IEEE Intelligent Vehicles Symposium*. 2010, pp. 560–565.

[32] E. Horster and R. Lienhart. «On the optimal placement of multiple visual sensors». In: *4th ACM intl workshop on video surveillance and sensor networks (VSSN)*. New York, NY, USA, 2006, pp. 111–120.

[33] J. Jung, C. Stachniss, and C. Kim. «Automatic Room Segmentation of 3D Laser Data Using Morphological Processing». In: *ISPRS International Journal of Geo-Information* 6.7 (2017).

[34] K. Kawashima et al. «Finding the next-best scanner position for as-built modeling of piping systems». In: *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, Volume XL-5*. 2014, pp. 313–320.

[35] T.P. Kersten et al. «Methods for Geometric Accuracy Investigations of Terrestrial Laser Scanning Systems». In: *Photogrammetrie Fernerkundung Geoinformation*. Vol. 4. 2009, pp. 301–314.

[36] M. K. Kim et al. «Optimal locations of terrestrial laser scanner for indoor mapping using genetic algorithm». In: *International Conference on Control, Automation and Information Sciences (ICCAIS)*. Dec. 2014, pp. 140–143.

[37] D.D. Lichti and S. Jamtsho. «Angular resolution of terrestrial laser scanners». In: *The photogrammetric record*. Vol. 21, pp. 141–160.

[38] E. Lindskog et al. «Combining point cloud technologies with discrete event simulation». In: *Proceedings of the 2012 Winter Simulation Conference (WSC)*. 2012, pp. 1–10.

[39] C. Liu, J. Wu, and Y. Furukawa. «FloorNet: A Unified Framework for Floorplan Reconstruction from 3D Scans». In: *European Conference on Computer Vision (ECCV)*. 2018.

[40] P. Lubos et al. «Touching the Cloud: Bimanual annotation of immersive point clouds». In: *IEEE Symposium on 3D User Interfaces (3DUI)*. 2014, pp. 191–192.

[41] K. Mamou and F. Ghorbel. «A simple and efficient approach for 3D mesh approximate convex decomposition». In: *16th IEEE International Conference on Image Processing (ICIP)*. 2009, pp. 3501–3504.

[42] K. Mechelke, T.P. Kersten, and M. Lindstaedt. «Comparative Investigations into the Accuracy Behaviour of the New Generation of Terrestrial Laser Scanning Systems». In: *Optical 3-D Measurement Techniques VIII*. Vol. 1. July 2007, pp. 319–327.

[43] Nicole Menck, Christian Weidig, and Jan C. Aurich. «Virtual Reality as a Collaboration Tool for Factory Planning based on Scenario Technique». In: *Procedia CIRP* 7 (2013), pp. 133 –138.

[44] M. Montemerlo, J. Becker, et al. «Junior: The Stanford entry in the Urban Challenge». In: *Journal of Field Robotics* 25.9 (2008), pp. 569–597.

[45] F. Moosmann, O. Pink, and C. Stiller. «Segmentation of 3D lidar data in non-flat urban environments using a local convexity criterion». In: *IEEE Intelligent Vehicles Symposium*. 2009, pp. 215–220.

[46] C. Mura et al. «Automatic room detection and reconstruction in cluttered indoor environments with complex room layouts». In: *Computers & Graphics* 44 (2014), pp. 20 –32.

[47] K. Na et al. «The ground segmentation of 3D LIDAR point cloud with the optimized region merging». In: *International Conference on Connected Vehicles and Expo (ICCVE)*. 2013, pp. 445–450.

[48] Hana Neradilova and Gabriel Fedorko. «Simulation of the Supply of Workplaces by the AGV in the Digital Factory». In: *Procedia Engineering* 192 (2017), pp. 638 –643.

[49] Takeru Niwa and Hiroshi Masuda. «Interactive collision detection for engineering plants based on large-scale point-clouds». In: *Computer-Aided Design and Applications* 13.4 (2016), pp. 511–518.

[50] S. Ochmann et al. «Automatic reconstruction of parametric building models from indoor point clouds». In: *Computers & Graphics* 54 (2016), pp. 94–103.

[51] S. Oesau, F. Lafarge, and P. Alliez. «Indoor scene reconstruction using feature sensitive primitive extraction and graph-cut». In: *ISPRS Journal of Photogrammetry and Remote Sensing* 90 (2014), pp. 68–82.

[52] B. Okorn et al. «Toward automated modeling of floor plans». In: *Proceedings of the symposium on 3D data processing, visualization and transmission*. Vol. 2. 2010.

[53] J. Pan et al. «Real-time collision detection and distance computation on point cloud sensor data». In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2013, pp. 3593–3599.

[54] M. Previtali et al. «Towards automatic indoor reconstruction of cluttered building rooms from point clouds». In: *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences* (2014), pp. 281–288.

[55] V. Sakenas et al. «Extraction of Semantic Floor Plans from 3D Point Cloud Maps». In: *2007 IEEE Intl Workshop on Safety, Security and Rescue Robotics (SSRR)*. 2007.

[56] V. Sanchez and A. Zakhor. «Planar 3D modeling of building interiors from point cloud data». In: *2012 19th IEEE International Conference on Image Processing (ICIP)*. 2012, pp. 1777–1780.

[57] G. A. Marcon dos Santos et al. «An adaptive algorithm for embedded real-time point cloud ground segmentation». In: *7th International Conference of Soft Computing and Pattern Recognition (SoCPaR)*. 2015, pp. 76–83.

[58] J. Schauer and A. Nüchter. «Removing non-static objects from 3D laser scan data». In: *ISPRS Journal of Photogrammetry and Remote Sensing* 143 (2018), pp. 15–38.

[59] Johannes Schauer and Andreas Nüchter. «Collision Detection Between Point Clouds Using an Efficient K-d Tree Implementation». In: *Adv. Eng. Inform.* 29.3 (Aug. 2015), pp. 440–458.

[60] Johannes Schauer et al. «Performance comparison between state-of-the-art point-cloud based collision detection approaches on the CPU and GPU». In: *IFAC-PapersOnLine* 49.30 (2016), pp. 54 –59.

[61] E. Shellshear, R. Berlin, and J. S. Carlson. «Maximizing Smart Factory Systems by Incrementally Updating Point Clouds». In: *IEEE Computer Graphics and Applications* 35.2 (2015), pp. 62–69.

[62] E. Shellshear, S. Tafuri, and J. Carlson. «A multi-threaded algorithm for computing the largest non-colliding moving geometry». In: *Computer-Aided Design* 49 (2014), pp. 1 –7.

[63] S. Soudarissanane and R. Lindenbergh. «Optimizing Terrestial Laser Scanning Measurement Set-up». In: *ISPRS workshop laser scanning*. Vol. XXXVIII. Calgary, Canada, 2011.

[64] S. Soudarissanane, R. Lindenbergh, and B. Gorte. «Reducing the error in terrestrial laser scanning by automatic optimization of the measurement set-up». In: *XXI ISPRS Congress, Beijing, China*. 2008.

[65] S. Soudarissanane et al. «Incidence angle influence on the quality of terrestial laser scanning points». In: *In Proceedings of Laserscanning*. Vol. XXXVIII. 2009, pp. 183–188.

[66] S. Soudarissanane et al. «Scanning geometry: Influencing factor on the quality of terrestrial laser scanning points». In: *ISPRS Journal of Photogrammetry and Remote Sensing*. Vol. 66. Jan. 2011, pp. 389–399.

[67] A. Stephan et al. «Interactive modelling of 3D-environments». In: *Proceedings. 11th IEEE International Workshop on Robot and Human Interactive Communication*. 2002, pp. 530–535.

[68] W. Sui et al. «Layer-Wise Floorplan Extraction for Automatic Urban Building Reconstruction». In: *IEEE Transactions on Visualization and Computer Graphics* 22.3 (2016), pp. 1261–1277.

[69] R. Tredinnick, M. Broecker, and K. Ponto. «Experiencing interior environments: New approaches for the immersive display of large-scale point cloud data». In: *IEEE Virtual Reality (VR)*. 2015, pp. 297–298.

[70] C. J. Turner et al. «Discrete Event Simulation and Virtual Reality Use in Industry: New Opportunities and Future Trends». In: *IEEE Transactions on Human-Machine Systems* 46.6 (2016), pp. 882–894.

[71] E. Turner and A. Zakhor. «Floor plan generation and room labeling of indoor environments from: Laser range data». In: *Proceedings of the 9th International Conference on Computer Graphics Theory and Applications (GRAPP)*. 2014, pp. 22–33.

[72] J. P. Underwood et al. «Explicit 3D change detection using ray-tracing in spherical coordinates». In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2013, pp. 4735–4741.

[73] C. Urmson, J. Anhalt, et al. «Autonomous driving in urban environments: Boss and the Urban Challenge». In: *Journal of Field Robotics* 25.8 (2008), pp. 425–466.

[74] V. Vavrík, M. Gregor, and P. Grznár. «Computer Simulation as a Tool for the Optimization of Logistics Using Automated Guided Vehicles». In: *Procedia Engineering* 192 (2017), pp. 923 –928.

[75] Y. Yao et al. «Can You See Me Now? Sensor Positioning for automated and Persistent Surveillance». In: *IEE Transactions on Systems, Man, and Cybernetics - Part B: Cybernetics*. Vol. 40. Feb. 2010, pp. 85–103.

[76] H. Yin, X. Yang, and C. He. «Spherical Coordinates Based Methods of Ground Extraction and Objects Segmentation Using 3-D LiDAR Sensor». In: *IEEE Intelligent Transportation Systems Magazine* 8.1 (2016), pp. 61–68.

[77] Keqi Zhang et al. «A progressive morphological filter for removing nonground measurements from airborne LIDAR data». In: *IEEE Transactions on Geoscience and Remote Sensing* 41.4 (2003), pp. 872–882.

[78] M. Zhang, D. D. Morris, and R. Fu. «Ground Segmentation Based on Loopy Belief Propagation for Sparse 3D Point Clouds». In: *International Conference on 3D Vision*. 2015, pp. 615–622.

[79] J. Zhao, S.C. Cheung, and T. Nguyen. «Optimal visual sensor network configuration». In: *Multi-Camera networks*. Vol. 6. May 2009, pp. 139–141.

[80] J. Zhao et al. «Approximate techniques in solving optimal camera placement problems». In: *IEEE Intl Conference on Computer Vision Workshops (ICCV Workshops)*. Nov. 2011.