Optimal positioning of terrestrial LiDAR scanner stations in complex 3D environments with a multiobjective optimization method based on GPU simulations

Gilles Rougeron, Jérémie Le Garrec, Claude Andriot

Optimal positioning of terrestrial LiDAR scanner stations in complex 3D environments with a multiobjective optimization method based on GPU simulations

Gilles Rougeron, Jérémie Le Garrec, Claude Andriot
Université Paris-Saclay, CEA, List, F-91120, Palaiseau, France

September 7, 2022

Abstract

Currently, the scanning of complex industrial sites is commonly performed using terrestrial LiDAR scanners. As the quality of the resulting point cloud depends mainly on the number and positions of LiDAR stations, this scanning process can be preliminarily optimized by means of a 3D model. A previous study proposed multiobjective optimization based on the linear scalarization of three functions to maximize coverage and overlapping of point cloud stations while minimizing their number. Because these objectives conflict, this study proposes the use of MO-CMA-ES, a global multiobjective optimization algorithm, to provide a full Pareto front and allow the user to make an informed decision. Our method is the first to rely on realistic LiDAR simulations that operate in fully 3D complex environments and provide point clouds with optionally noisy coordinates. For performance considerations, ray-traced simulations and objective evaluations were performed using a GPU. Furthermore, clash detection in the proximity of station positions was also considered. After validating our method's behavior and demonstrating its superiority over the conventional approach in a simple case, we conducted a study on an industrial-grade case based on a 2.7-million-triangle model, further demonstrating our method's effectiveness by producing a minimal 15-station solution with optimal coverage and overlapping.

Keywords— Terrestrial LiDAR, Survey Design, 3D Point Clouds, Multiobjective Problem, Simulation Modeling, GPU Computing.

1 Introduction

As part of the digital transition related to Industry 4.0, the creation of digital twins at equipment and industrial sites is becoming increasingly widespread. Digital twins can be used for a wide range of purposes, from simple visualization to verification and validation of concepts, progress monitoring, conformity measurements, or the training of new staff. To maximize accuracy, it is necessary to carry out 3D surveys of the geometry and appearance of installations. Among the many types of sensors and techniques developed for this purpose, terrestrial LiDAR scanners (TLS) remain the sensors of choice due to their high precision, high point density, and long range.

In general, owing to their complex geometries, it is necessary to conduct surveys at multiple positions, called LiDAR stations. The scans are then postprocessed, denoised, colorized, registered, and merged. This process generates a potentially massive 3D point cloud that can be analyzed by means of a screen or virtual reality (VR) headset. It can also be used for different operations such as:

- measurement of distances to a previous point cloud or digital model as a component of progress or conformity tests,
- triangular meshing to obtain a lighter geometrical representation better suited to collision detection for instance,
- segmentation and idealization to reach higher-level geometric representations, such as canonical shapes, BREP-type CADs, or BIM models, in the process of scan2BIM,
- object detection and 6DoF pose estimation of known CAD parts.

These operations can be performed using classical methods or deep learning-based approaches [12].

The ease of exploitation of the resulting point cloud is directly related to its quality. In particular, occlusions must be avoided. Theoretically, every element of the site, including all equipment, should be visible to at least one of the stations. Point density must therefore be as homogeneous as possible, and reach a minimum threshold in the largest possible area. Precise registration requires overlap between point clouds issued from different stations. The noise level must be as low as possible. And finally, to minimize the amount of raw data to process and avoid breaching potential time constraints, the number of stations should be minimal.

Today, decisions relating to the necessary amount and placement of stations depend on the knowledge and experience of operators. In this study, we propose a novel approach for designing optimal solutions through a multiobjective optimization process. Given a 3D model of a site or equipment to scan, along with its surrounding environment and a 3D-accessible area for TLS stations, sets of LiDAR network simulations with fixed parameter values were performed to maximize the quality of surveyed point clouds through different criteria (coverage and overlap) while minimizing the number of stations. A Pareto front was characterized by these conflicting objectives to be used by an expert for informed decision-making.

Our main contributions to the LiDAR optimal positioning problem are:

- A multiobjective optimization approach is formulated in a continuous search space to provide rich resulting information;
- A method based on fast and realistic LiDAR simulations performed in a complex 3D world;
- Objectives evaluation is entirely performed on GPU;
- Solution accounts for clash detection between LiDAR stations and the 3D model.

To the best of our knowledge, this is the only method that addresses the problem in full 3D space and evaluates point cloud quality based on simulations. Our solution can be used indoors, outdoors, or in a combined environment.

After a brief review of related studies in Section 2, we present the LiDAR simulation tool in Section 3. Section 4 provides a description of the 3D model's components, surveyed and blocking meshes, and the search area, followed by a presentation of the continuous search space, objective evaluations, and multiobjective strategy used. Section 5 provides the results for simple cases as a validation and characterization of the method's performance followed by an analysis of complex case results. Finally, after drawing some perspectives in Section 6, we conclude the paper in Section 7.

2 Previous work

The "art gallery problem" is a fundamental problem that entails minimizing the number of guards that collectively observe a whole art gallery. This problem has been the subject of numerous theoretical studies and practical applications, the most important of which relate to view-planning. The problem can be approached in two ways depending on whether an environmental model is available. In the model-free case, space is discovered through observation, and next best-view solutions (NBV) are commonly applied in an online setting. By contrast, sensor network design is generally performed offline for cases with a known model. Recent surveys, dedicated to the camera placement problem in surveillance applications [21] and the reconstruction of 3D CAD models of objects [26], demonstrate that most proposed solutions boil down to solving an NP-hard set cover problem (SCP), which entails finding the smallest sequence of positions that covers a space.

For TLS networks, the authors of [2] raised the problem of planning for scanning (P4S) and reviewed the most recent studies. The specificity of P4S relates to the sensor characteristics and criteria used to qualify a 3D point cloud, including coverage completeness, accuracy, spatial density, and registrability. Most previous studies [32, 19, 5] operated in a 2D environment, where planar walls were simplified to line segments, and the search area was discretized in a set of possible positions. These studies generally employed a simple or modified greedy approach, which, although suboptimal, provides reasonable local optimal solutions to the SCP problem. This approach locates an initial position with the highest coverage, and successively adds new positions with the highest additional contributions until the global estimated quality of the point cloud stops improving. Surface visibility, point density, accuracy, and overlap are estimated through geometric considerations such as the line of sight, distance from position, and incident angle to segments. Interactive approaches in which an expert contributes to the greedy optimization process were proposed in [1]. In [9], the authors implemented a greedy approach on a GPU with both coverage and overlap estimations. In [19], LiDAR and artificial target positions used to ease registration were optimized using a hierarchical strategy followed by a modified greedy method.

Of the few studies [31, 39, 25] that operated in a 3D environment, the last used a voxel grid to estimate coverage. Its search area was a 2D regular planar grid with candidate positions, possibly at different heights.

In [22], a deep reinforcement learning (DRL) approach was used to locate the optimal set of view poses allowing the scanning of a 3D CAD model with a depth camera sensor placed on a robotic arm. An agent attempted to maximize its reward by finding successive positions that yielded the highest improvement in coverage, until a given maximal number of poses was reached. The covered surface area was estimated using a noisy camera simulation performed on a GPU, and a voxelized representation of the 3D CAD model. Several DRL algorithms were compared for both discrete and continuous search spaces. Much like the greedy approach, this method works well in practice but yields suboptimal results.

Instead of progressively adding sensors, several studies approached the problem by optimizing the global network sensor positions while minimizing their number. In [6], in the context of indoor environment surveying, a set of candidate points was distributed over a floorplan map with a 2D discrete grid, and the coverage problem was solved with an exact approach via integer linear programming (ILP). By constraining the connectivity of the LiDAR network graph, overlapping is solved using mixed integer linear programming (MILP). Although this approach incurs an exponential complexity in the worst case, it is generally applicable in practice.

Other studies employed heuristic-based optimization methods to explore the entire search space, and did not exhibit the worst-case problem. The present study falls under this category. In the field of video cameras, the authors of [35] proposed a two-step method. Starting from an initial huge set of cameras with given positions, orientations, and parameters, providing almost perfect coverage, the first step consists of finding the minimal subset allowing 80% of the maximal observed coverage to be reached. In the second step, the positions and orientations of a given number of cameras are allowed to move in a continuous search space to improve coverage. Both optimization steps rely on the modified particle swarm optimization (PSO) method. For TLS networks, in [20], a genetic algorithm (GA) was used that finds a network with numerous positions providing full coverage, and gradually removed positions until a minimal threshold is reached. In [18], the performances of GA, PSO, and simulated annealing (SA) were compared in a single given environment. Both aforementioned studies operated in a 2D world with positions spread over a discrete area.

In [4], the authors proposed the simultaneous optimization of coverage and overlapping while minimizing the number of stations. This multiobjective optimization is performed by a GA method with fitness value F for a given network of stations s, resulting from the linear scalarization of three functions, as expressed by Equation 1.

$$F(s) = \lambda_1\Gamma(s) + \lambda_2\Phi(s) + \lambda_3\Delta(s) \tag{1}$$

where:

- $\Gamma(s)$ denotes the coverage value.
- $\Phi(s) = 2^{\frac{N_{max}-N_{stations}}{N_{max}}} - 1$ is a function that decreases with an increase in $N_{stations}$, the number of active stations for a given network, given all networks having N_{max} stations,
- $\Delta(s) = \frac{1}{Cc}$ where Cc is the number of connected components in the LiDAR station visibility graph.
- $\lambda_1 = \lambda_2 = 0.4$ and $\lambda_3 = 0.2$

Because all three functions have a range of $[0, 1]$ and the sum of weights λ_i is 1, F is also in the range of $[0, 1]$. The GA seeks to maximize this fitness function by evolving a population of station networks in a 2D continuous search space. One can remark that the optimal solution is dependent on the three function formulations and weights, as changes in any of these yield a new optimal solution. By contrast, the multiobjective approach described in this paper provides the user with a Pareto front, allowing the solution to better match the requirements. Furthermore, our approach is based on realistic 3D LiDAR simulations that account for considerably more complex environments.

3 LiDAR simulation

3.1 Related studies

Many LiDAR simulation tools are available today, and are commonly used to test the performance of new sensors, benchmark point-cloud algorithms, and create rich datasets for deep learning methods. In robotics, they can be used for sensor-based motion planning [28]. For autonomous vehicle simulation, they enable the verification of whether fixed or moving obstacles can be efficiently detected (see [7] and [27]). In these fields, speed is generally favored to realism, as there is a need for online simulations. In [24], the authors present a method that accounts for the rolling-shutter effect of a moving vehicle carrying LiDARs, and the motion blur of moving obstacles.

However, simulation tools for time-of-flight terrestrial LiDAR are often used offline, and they range from simple to physically-based models. Blensor [11] was an early attempt to create a tool integrated into the Blender software that relies on ray casting. An implementation of a similar tool with GPU computation using Nvidia Optix was

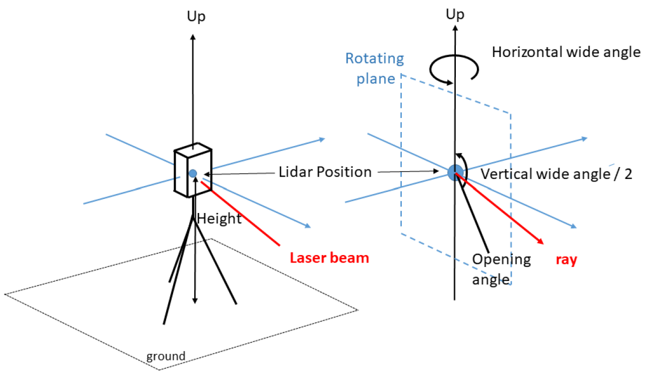


Figure 1: LiDAR station geometry.

reported in [23]. More realistic models were implemented by Helios [37], which can consider the laser beam divergence and emitted signal waveform. This allows for the detection of multiple points in a single direction, corresponding to multiple signal peaks, for time-of-flight LiDARs. Finally, in DIRSIG [10], a physical simulation code was implemented based on complex models of source, transport, light with material interaction, and receiver.

Because multiobjective optimization requires many simulations of station networks, where tens of millions of points are recorded for each station, we decided to implement our tool on the GPU to reach an acceptable computation time. Our method is based on a simple ray-casting approach, and provides rich results that are further utilized on a GPU to rapidly evaluate objective values. Realism can be improved by adding noise.

3.2 Our LiDAR simulation tool

Simulation inputs To scan the surrounding environment, a terrestrial LiDAR is generally placed on a tripod under several geometric parameters, as shown in Figure 1. Once these parameters are set, the survey is initiated. A rotating mirror enables the vertical deviation of the laser beam around the station in a plane, while a step-by-step engine progressively rotates this plane. Globally, the survey samples an almost perfectly spherical region around the station position.

The LiDAR simulation input parameters included the following:

- Positional coordinates
- Upward direction, assumed to be vertical
- Height, or distance from position to floor
- Horizontal wide angle (generally 360°)
- Vertical wide angle (generally less than 360° with an angle opening downward)
- Horizontal and vertical step angles
- Distance range $[d_{min}, d_{max}]$

While simulating a network of LiDAR stations, all parameters were fixed with the exception of LiDAR positions. The geometrical input model of the simulation tool consisted of triangular meshes with vertex normals. These meshes can be derived from a BREP CAD model or previous coarse point cloud scan. They can transform by means of translation, rotation, and scaling, and have attached materials that contain a diffuse reflection coefficient.

Simulation The implemented LiDAR model employed ray casting. The code was based on the Nvidia Optix 7 library, allowing access to the power of ray tracing hardware support. The LiDAR simulation tool was integrated into the Unity 3D platform.

A ray was launched for each laser beam direction. When the ray hit a surface, the following results were obtained (see Figure 2):

- *point position*, or the first intersection point identified. It is computed in world space but can be transformed into a local LiDAR coordinate system.
- *point color*: An equirectangular panoramic image is first rendered around the current station and transferred to the GPU as a texture. Then, during the LiDAR simulation, for each ray direction hitting a surface, a point color is obtained by bilinear texture color interpolation.
- *point normal*, obtained by bilinear interpolation of the hit triangle's vertex normals with (s, t) barycentric coordinates of intersected points. It is oriented toward the direction of the incident ray.
- *reflectivity*: Following Lambertian law, it is equal to $R(I \cdot (-N))$, where R is the diffuse coefficient of the material associated with the intersecting mesh, I is the direction of the incident ray, N is the point normal, and $\cdot$ is the dot product operation
- *mesh id*: The ID of the triangle's mesh, which remains at -1 if no hit occurs.

The computations of point color, normal, and reflectivity are optional. On the GPU, different output buffers were pre-allocated to fit the number of rays.

3.3 Noise

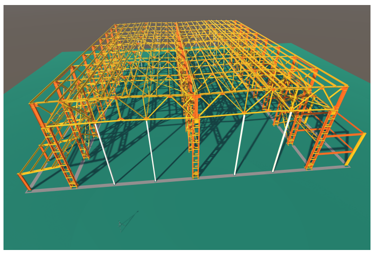
Noise model The geometric noise of a point cloud may have several origins, such as electronic noise on the sensor, interaction with the atmosphere, or complex interaction of the laser beam with a surface. In this study, we conducted realistic but not physically-based LiDAR simulations. Therefore, we employed tables provided by LiDAR constructors, thus allowing the evaluation of noise amplitude for perfectly accurate measures (see Table 1). As can be seen, noise amplitude increases with higher distance and lower reflectivity. By using a Lambertian diffuse law to evaluate reflectivity (see Section 3.2), we reached global agreement with the increase in the incident angle to the intersection point normal, as noted in [33].

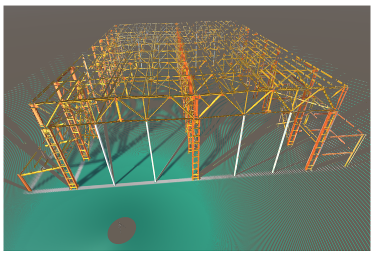
Reflectivity \Distance	5m	10m	20m	40m	60m
Black 8%	0.5mm	0.6mm	0.7mm	2.5mm	5.0mm
Grey 21%	0.4mm	0.5mm	0.6mm	0.8mm	2.0mm
White 89%	0.3mm	0.4mm	0.5mm	0.6mm	1.0mm

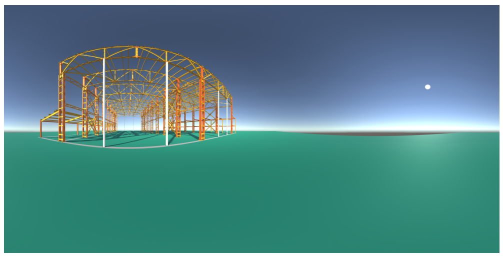
Table 1: Noise amplitude of Leica RTC 360 for a single measurement, depending on the distance and reflectivity of the obstacle.

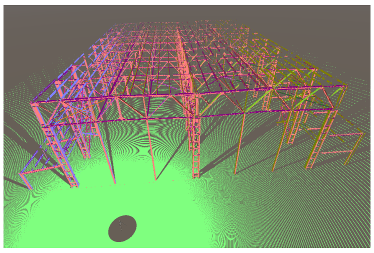
Noise computation method During simulation, each intersection point was randomly moved along the ray following a zero-mean Gaussian distribution, in which the standard deviation σ value was computed by interpolating the values of the constructor amplitude table.

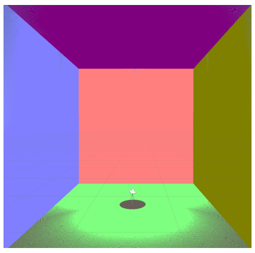
For computational performance, this initial table was transformed via interpolation and extrapolation into a regular oversampled table. Its inputs were normalized

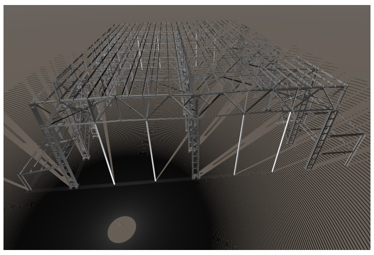

(a) A 3D mesh model with a terrestrial LiDAR positioned in front of it.

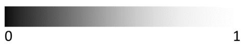

(b) Point cloud colors.


(c) Rendered panoramic image from LiDAR position.

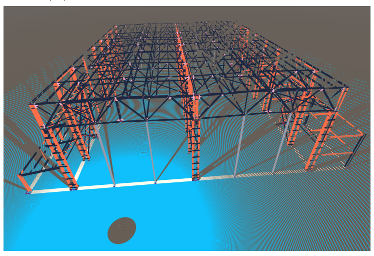

(d) Point cloud normals in false colors.

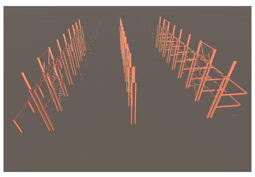

(e) Normals color map. A cubic model was internally scanned using LiDAR.


(f) Point cloud reflectivity.



(g) Reflectivity grayscale.


(h) Point cloud mesh IDs in false colors.


(i) Model mesh ID 6.

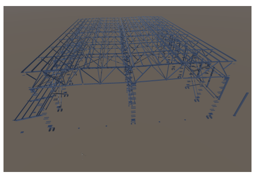

(j) Model mesh ID 7.

Figure 2: Simulated LiDAR scan. A model with dimensions of 65.5 m x 50 m x 15.5 m was placed on a 100 m x 100 m surface. The model contains 17 meshes.

distance and reflectivity, both in the range of $[0, 1]$. We chose a sampling step of 0.01, which allowed the creation of a 101×101 table that was transferred as a texture map to the GPU for fast σ evaluation by bilinear interpolation. Algorithm 1 summarizes the methods used.

Algorithm 1 Noise computation. Operations in blue are performed on GPU.

```
{PRE-COMPUTATION}
From initial constructor table, create a regular oversampled noiseTable
Transfer it as texture map to GPU
{COMPUTATION}
{[$d_{min}$, $d_{max}$] is the LiDAR range}
{$I$ is the incident ray direction; $R$ is the diffuse reflection coefficient}
for each intersectionPoint found do
    distance ← Norm(lidarPosition, intersectionPoint)
    normalizedDistance ← $\frac{distance - d_{min}}{d_{max} - d_{min}}$
    $N$ ← InterpolatedAndOrientedNormalAt(intersectionPoint)
    reflectivity ← $R(I \cdot (-N))$
    $\sigma$ ← BilinearInterpolation(noiseTable, normalizedDistance, reflectivity)
    $\epsilon$ ← RandomValueWithGaussianDistribution$(0, \sigma)$
    intersectionPoint ← intersectionPoint + $\epsilon I$
end for
```

4 Optimization of LiDAR positioning

The general positioning optimization method relies on multiobjective optimization based on LiDAR simulations computed using the proposed tool. The elements of the method and their associated algorithms are presented in the following sections: surveyed and blocking meshes in Section 4.1, search area in Section 4.2, clash detection around LiDAR stations in Section 4.3, and search space in Section 4.4. The three evaluated objectives are introduced in Section 4.5: coverage, overlap, and station number. The basic principles of the chosen optimization methods – CMA-ES for mono-objective problems and MO-CMA-ES for multiobjective problems – are presented in Section 4.6.

4.1 Surveyed and blocking 3D triangular meshes

When scanning a complete site, the entire 3D model must be covered. However, specific equipment and areas of complex industrial environments are often surrounded by elements that limit their visibility and accessibility. For instance, a machine tool may be surrounded by other machines or architectural elements, such as barriers or posts. In the context of our application, the user specifies the part of the model to be scanned. These models are referred to as surveyed meshes, whereas the remaining models are referred to as blocking meshes. Although both mesh types can intersect rays and be used for clash detection, only the intersection points of rays with surveyed meshes contribute to coverage and overlapping objectives (see Section 4.5).

During the initialization step, we voxelized the surveyed meshes' surfaces by implementing the method described in [30] on a CPU. We used a cubic box whose length corresponds to the largest extent of the bounding box of all surveyed meshes. Given the size provided by the user, the voxel size is defined as

$$voxel\_size = \frac{box\_length}{2^d} \tag{2}$$

where d is the smallest integer such that $voxel\_size \leq user\_size$. It should be noted that the user size should be selected so that voxelization is really surfacic. For instance, it should be smaller than the wall thickness if both faces are visible. However, the voxel size must not be too small, as explained in Section 4.5.1.

Any voxels that intersect triangles are stored in memory with an associative map, whose keys are the Morton keys of an octree with constant depth d. This list of full voxel keys (referred to as surveyed geometry voxels throughout this paper) is sorted and stored in a GPU memory buffer.

4.2 Search area

The search area is the accessible area where LiDAR stations can be positioned. Its surface is not intersectable, although it often coincides with a blocking surface (e.g., the ground), and it is a 3D triangular mesh with u_i, v_i coordinates attached to its vertices. The search area can be represented by any parametric surface, and may contain slopes, curves, bumps, holes, and similar features. The triangular mesh may even have different connected components to represent a set of surfaces (the different stairs of a building, for instance) as long as the u_i, v_i coordinates of the vertices do not overlap.

Optimization occurs as LiDAR stations evolve in this search area. Provided a vertical height and 2D coordinate point (u, v) with continuous values, a given station's position can be computed on the CPU by finding a triangle with vertex i, j, k and coordinates u_i, v_i, u_j, v_j, u_k and v_k such that

$$\begin{aligned} u &= (1-s-t)u_i + su_j + tu_k \\ v &= (1-s-t)v_i + sv_j + tv_k \end{aligned} \tag{3}$$

where s and t are the barycentric coordinates of a point within the triangle such that

$$0 \leq s \leq 1,\ 0 \leq t \leq 1 \text{ and } 0 \leq s+t \leq 1 \tag{4}$$

To efficiently search this triangle and compute values of s and t, a quadtree is built on the triangular mesh of the search area in the 2D (u, v) space. The quadtree spans $[u_{min}, u_{max}]$ and $[v_{min}, v_{max}]$, corresponding to the overall search coordinates, and its root node contains all mesh triangles. The build is recursive, as each node is subdivided into four, and each triangle is distributed into one or more child nodes. The process terminates for a given node when the number of triangles becomes smaller than the maximum allowed, or a given maximum depth is reached. During the triangle search, this enables the quick pruning of nodes that do not contain (u, v) coordinates. For the remaining nodes, the searched triangle, if it exists, is found by inverting Equation 5 and ensuring that s and t obey Equation 4. If no triangle is found, a hole is present, in which case the LiDAR station is not simulated and does not contribute to the objective evaluation. The inverse equations are as follows:

$$\begin{aligned} s &= \frac{(v-v_i)(u_k-u_i) - (u-u_i)(v_k-v_i)}{(v_j-v_i)(u_k-u_i) - (u_j-u_i)(v_k-vi)} \\ t &= \frac{(u-u_i) - (u_j-u_i)s}{u_k-u_i} = \frac{(v-v_i) - (v_j-v_i)s}{v_k-v_i} \end{aligned} \tag{5}$$

If we denote the triangle vertices as P_i, P_j, and P_k, then the LiDAR position in 3D is

$$lidarPosition(u, v) = (1-s-t)P_i + sP_j + tP_k + h\ up \tag{6}$$

where h is the LiDAR height above the ground and up the vertical direction in world coordinates. In our case, all terrestrial LiDARs were assumed to remain vertical. In other contexts, a 3D triangular local normal mesh may be used.

4.3 Clash detection

LiDAR stations should not be too close, or even intersect the surveyed or blocking geometries. In practice, an on-site operator should be able to move in the proximity of the TLS tripod. Therefore, a cylindrical security volume is used around each station. For simplicity, this volume is approximated by a set of two spheres, as shown in Figure 3. The first sphere is at height h_1 above the ground, whereas the second is at height h_2. Both spheres have a radius of r. To avoid contact with the ground, r should remain smaller than h_1. Using the Embree library from Intel, an acceleration structure is created on a CPU containing surveyed and blocking meshes. For each LiDAR station, the minimal distance from each sphere center C_1 and C_2 to the meshes is computed using an *rtcPointQuery* Embree function call. If one of the minimal distances is smaller than the sphere radius, the station simulation is skipped and does not contribute to the objective evaluation.

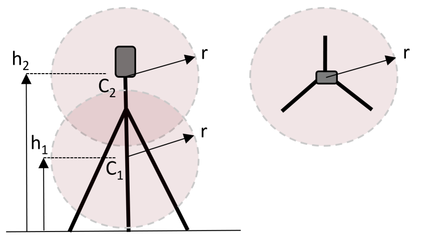


Figure 3: Security volume around each station. Left is front view, right is above view.

4.4 Search space

The search space is the space in which an optimization algorithm operates. If the number of stations N is given, as in a FIX-type problem, its dimension is $2N$, as each candidate solution of the population, or network of LiDAR stations, is represented by a coordinate vector $(u_1, v_1, ..u_N, v_N)$. The LiDAR positions in 3D are computed on the CPU for each station i $(1 \leq i \leq N)$ using the method described in Section 4.2.

If optimization also attempts to minimize the number of stations, as in a MIN-type problem, then each candidate solution has a number of valid stations $N_{stations}$ between two user-provided values N_{min} and N_{max}, which can be roughly estimated based on the extent and complexity of surveyed geometry. Because the optimization libraries used in our implementation (see Section 4.6) do not allow the mixing of floating-point and integer-value coordinates, a floating value coordinate α between 0 and 1 is added to all candidate solutions, resulting in a search space of dimension $2N_{max} + 1$. The number of stations evaluated for a given candidate is

$$N_{stations} = rounded(N_{min} + \alpha(N_{max} - N_{min})) \tag{7}$$

If the (u, v) coordinates in the search area are bounded by $[u_{min}, u_{max}]$ and $[v_{min}, v_{max}]$, then the optimizations are constrained by the following bounds:

- $[(u_{min}, v_{min}, ..u_{min}, v_{min}), (u_{max}, v_{max}, ..u_{max}, v_{max})]$, two $(2N)$ vectors for a given number of stations,
- $[(u_{min}, v_{min}, ..u_{min}, v_{min}, 0), (u_{max}, v_{max}, ..u_{max}, v_{max}, 1)]$, two $(2N_{max}+1)$ vectors in case we seek to minimize the number of stations.

4.5 Objectives

To assess the quality of scans provided by each candidate solution, up to three parameters were evaluated. The following objectives must be optimized.

4.5.1 Coverage

Coverage was measured through local point density estimation in each surveyed geometry voxel cell. The user provided a density value, generally expressed as the point spacing every n mm. Assuming that points were locally distributed on a regular grid, which implies a surfacic density of 1 point every $n^2 mm^2$, and that voxel volumes were locally intersected by a single planar surface with area $voxel\_size^2$, a $goal\_density$, representing the number of points per voxel with floating point values, can be computed as the ratio $\frac{voxel\_size^2}{n^2}$. After each simulation, the surveyed geometry voxel point count $voxel\_density$ (denoted voxelDensityBuf in Algorithm 2) increased by the number of points it contained. Finally, after all stations were simulated, coverage was estimated using the following formula:

$$Coverage = \frac{\sum_{sg\_voxels} min(\frac{voxel\_density}{goal\_density}, 1)}{N_v} \tag{8}$$

where N_v denotes the total number of surveyed geometric voxels (sg_voxels). It should be noted that $voxel\_size$ must exceed the point spacing n so that the required number of points per voxel $goal\_density$ is at least 1.

Coverage value is in the $[0, 1]$ range, where the maximal value of 1 corresponds to a situation where all surveyed voxel densities have reached the density goal. Coverage measures local densities as well as the visibility of global surveyed geometry. This formula naturally forces the optimization method to spread stations throughout the search area. Because the ratio $\frac{voxel\_density}{goal\_density}$ is limited to 1, it penalizes solutions with useless local over-densities that result from an excessive number of stations covering the same area. Furthermore, as point density decreases with distance to the LiDAR position and grazing angles, the station positions should not be too far from the surveyed geometry, or at too large angles of incident rays with normals.

Coverage was computed on a GPU with CUDA Thrust code without transferring point cloud data to CPU RAM. Each point coordinate was converted into a Morton key value that was binary searched in the surveyed geometry voxel key buffer. Subsequently, the point keys were counted through a histogram algorithm, resulting in the addition of a point count per voxel. Once the simulations of all LiDAR stations were completed, coverage was computed using a simple additive reduction of the threshold density ratios. A 3D cartography can be exported by visualizing these density ratios using false colors.

4.5.2 Overlap

After a scanning survey of a real site, the point clouds of each LiDAR station are registered iteratively or globally (see [15] for a recent survey regarding this topic). When no artificial targets are added, high overlapping between scans improves the precision of this process by increasing the number of feature-matching pairs.

The first step of the overlap estimation method for a given candidate solution consists of evaluating the overlapping of each of its two stations i and j. To accomplish this, a list of stations is determined for each surveyed geometry voxel. It should be noted that a simple condition for voxel observation is to contain at least one point from the simulated point cloud of a given station. In practice, to preserve memory and maintain GPU efficiency, instead of building a real list, a 64-bit long integer $voxel\_stations\_ID$, initially set to 0, is associated with each surveyed geometry voxel[1]. This value is denoted by voxelStationsIDBuf in Algorithm 2. For each point within a point cloud, after determining the surveyed geometry voxel key that contains it, $voxel\_stations\_ID$ is modified as follows:

$$voxel\_stations\_ID \leftarrow voxel\_stations\_ID \; OR \; lbs(1, i-1) \tag{9}$$

where i is the station index in $[1, N_{stations}]$, and lbs is the left-bit shift operation. The visibility of a given surveyed geometry voxel by station i can then be represented by the variable $voxel\_visible_i$, whose value is 1 if

$$voxel\_stations\_ID \; AND \; lbs(1, i-1) \neq 0 \tag{10}$$

and 0 otherwise.

The overlapping of stations i and j is evaluated using

$$Overlap_{i,j} = \frac{\sum_{sg\_voxels} voxel\_visible_i \; AND \; voxel\_visible_j}{\sum_{sg\_voxels} voxel\_visible_i \; OR \; voxel\_visible_j} \tag{11}$$

with a real value in the range of $[0, 1]$, where 0 corresponds to no overlap at all, and 1 corresponds to a perfect overlap.

Because of the symmetry of the formula $Overlap_{i,j} = Overlap_{j,i}$, and because $Overlap_{i,i} = 1$, only the $\frac{N_{stations}(N_{stations}-1)}{2}$ overlap couples have to be evaluated.

This generates an undirected weighted graph in which the adjacency matrix coefficients are $Overlap_{i,j}$, with the exception of $Overlap_{i,i}$, which are set to 0 because, by convention, nodes are not connected to themselves. We evaluated the overlap of a candidate solution as the average clustering coefficient of this graph. This coefficient indicates the degree to which graph nodes are related. Intuitively, given a node triplet (i, j, k), the coefficient indicates the probability that j and k are connected assuming i is connected to both j and k.

For an unweighted graph with N nodes, where all adjacency coefficients $A_{i,j}$ are either 0 or 1, the average clustering coefficient is

$$\begin{aligned} C &= \frac{1}{N}\sum_{i=1}^{N} C_i \\ C_i &= \frac{1}{k_i(k_i-1)} \sum_{j,k} A_{i,j}A_{i,k}A_{j,k} \end{aligned} \tag{12}$$

where k_i is the degree of node i.

Several generalizations of weighted graphs have been proposed (see [36]). We decided to use the formula suggested in [29] because it is fast and numerically stable. It also presents desirable properties, such as general versatility, continuity with the unweighted formula, and robustness to noise. This leads to the following equation:

$$\begin{aligned} Overlap &= \frac{1}{N_{stations}} \sum_{i=1}^{N_{stations}} Overlap_i \\ Overlap_i &= \frac{1}{max_{i,j}(Overlap_{i,j})} \frac{\sum_{j,k} Overlap_{i,j}Overlap_{i,k}Overlap_{j,k}}{\sum_{j,k,j\neq k} Overlap_{i,j}Overlap_{i,k}} \end{aligned} \tag{13}$$

which is the overlap for a given candidate solution with a value in the range of $[0, 1]$.

It should be noted that, because the notion of the clustering coefficient relies on triplets of station connections, it contributes to the possible detection of numerous feature matching pairs between point clouds. This should ensure robust and precise global registration.

[1]N_{max} can therefore not exceed 64 in our current implementation.

In our implementation, voxel station visibility was evaluated on the GPU after each LiDAR simulation. The adjacency matrix coefficients were computed on the CPU by dividing the results of additive reductions performed on the GPU. The final formula (13) was evaluated on the CPU.

4.5.3 Number of stations

In an MIN problem, the objective that minimizes the number of stations is evaluated for each candidate solution as the value of its α coordinate. This value is bounded by 0 and 1, where 0 corresponds to N_{min} and 1 corresponds to N_{max} (see Equation 7).

4.5.4 Optimization problem types

A global optimum must be found in the search space that corresponds to a maximum for coverage and overlap, and a minimum for α (i.e., the number of stations). As these objectives are all in the $[0, 1]$ range, the problem is converted into the minimization of $1 - Coverage$, $1 - Overlap$, and α.

Instead of always optimizing these three objectives, we addressed four optimization problems that correspond to different combinations: COVERAGE_FIX, COVERAGE_OVERLAP_FIX, COVERAGE_MIN, and COVERAGE_OVERLAP_MIN. Table 2 presents the characteristics of these problems.

Optimization problem	Objectives to minimize			Optimization type	
	1 - Coverage	1 - Overlap	α	Mono-objective	Multiobjective
COVERAGE_FIX	✓			✓	
COVERAGE_OVERLAP_FIX	✓	✓			✓
COVERAGE_MIN	✓		✓		✓
COVERAGE_OVERLAP_MIN	✓	✓	✓		✓

Table 2: Different optimization problem types addressed.

COVERAGE_FIX is presented in this paper primarily for pedagogical purposes, whereas the other three problems provided the most interesting results. In practice, as discussed in Sections 5.1.2 and 5.2, for a given test case, a variety of multiobjective optimization steps are performed to efficiently explore the search space. Decisions to guide the search and select the best compromise must be made after each step based on user criteria.

It should be noted that multiobjective optimization often includes conflicting objectives. In the context of this study, an increase in coverage requires a large number of LiDAR stations to be spread in the search area, while an increase in overlap is associated with grouping. For MIN optimization, the number of stations should be reduced.

4.6 Optimization methods

For the COVERAGE_FIX problem, we decided to use CMA-ES [13], which is a powerful global mono-objective, derivative-free method based on an evolutionary strategy. Each iteration updates the covariance matrix of a multivariate normal distribution in $\mathbb{R}^{dim}$, where dim is the dimension of the search space. This is highly efficient because it enables learning from a second-order model of the underlying objective function, which is similar to the approximation of the inverse Hessian matrix in the quasi-Newton method in classical optimization. See [14] for a comparison with other optimization methods on a benchmark. In addition, this approach has very few parameters to set, namely the number of iterations, population size, and standard deviation sigma initial value[2].

For the last three problems listed in Table 2, many multiobjective optimization methods can be used [8], among which the most common are NGSA II, NGSA III, and MOEA/D. Nevertheless, we selected MO-CMA-ES ([34], [17]), as it inherits the benefits of its mono-objective version, including a rapid convergence rate and the same number of required parameters. Furthermore, it uses the hypervolume between the Pareto front and a reference point (the unit vector in our case) as a performance indicator. This proves useful when it comes to knowing when to stop iterating, as it tends to converge asymptotically to a maximum value. It should be noted here that as the variables we seek to minimize ($1 - \textit{coverage}$, $1 - \textit{overlap}$, and α) all have minimal values of 0, the maximal theoretical hypervolume corresponds to a unit square in the case of bi-objective optimization (COVERAGE_OVERLAP_FIX and COVERAGE_MIN), and a unit cube in the case of tri-objective optimization (COVERAGE_OVERLAP_MIN).

We used libcmaes [3] for the implementation of CMA-ES, and Shark [16] for that of MO-CMA-ES. The optimizations were constrained as described in Section 4.4. For CMA-ES, the initial population was derived from a single network of initial LiDAR positioning provided by the user, whereas for MO-CMA-ES, it was randomly generated in the search space.

Algorithm 2 globally sums up the LiDAR positioning optimization method employed in this study.

5 Results

5.1 Simple cases

5.1.1 Noise impact

In this first test, a LiDAR was positioned in front of three surveyed walls represented by simple rectangles (see Figure 4). The wall material was assigned a diffuse-coefficient value of 0.1. Two LiDAR simulations followed by coverage estimation were performed without noise, and then performed again with noise. We used the noise parameters listed in Table 1, and standard deviations ranged from 0.5 to 2 mm at normal incidence. The user and voxel sizes were set to 1 cm. Point spacing was set at 1 point every 6 mm, which implies a goal density of 2.8 points per voxel.

	wo noise	w noise
number of points	383 659	383 659
number of points on surface	383 659	371 637
Coverage	0.308	0.299

Table 3: Evaluation results.

Table 3 indicates that the number of points found on the wall surface (i.e., in the surveyed geometry voxels) in the presence of noise was smaller than the total number of points. Because outliers do not contribute to coverage and overlap objective values, simulation coverage with noise was slightly lower than that without noise. Figures 5a and 5b denote outliers in red. Although the point density is smaller on distant walls, these outliers are proportionally more abundant.

Adding noise to LiDAR simulations for positioning optimization introduces an effect that tends to reinforce global trends. Solution candidates with a high noise level – that is, with LiDAR positions too far away from the surveyed geometry, or

[2]For the results presented in Section 5, this value was set to 0.5.

Algorithm 2 General Multiobjective Optimization Algorithm.
Operations in blue are performed on GPU.

```
{PARAMETERS}
{lidarParameters, surveyed/blocking meshes, mesh materials, computeNoise}
{$userSize$, $pointSpacing$, search area mesh, $h$ LiDAR height}
{$h_1, h_2, r$ security volume parameters around LiDAR}
{MO-CMA-ES parameters: M population size, nbIterMax number of iterations, and sigma initial standard deviation}
{$N$ number of stations for FIX optimization, $[N_{min}, N_{max}]$ for MIN optimization}
{INIT}
Init raytracing (Nvidia Optix pipeline and acceleration structure), allocate Cuda buffers: point cloud positions, colors, normals, reflectivities and mesh ids
if computeNoise then
  Init oversampled noiseTable and transfer it onto GPU (§3.3)
end if
$voxelSize \leftarrow$ ComputeVoxelSize(surveyed meshes, $userSize$) (§4.1)
Voxelize surveyed meshes and transfer result as GPU Cuda Buffer voxelKeysBuf
Allocate cuda buffers voxelDensityBuf, voxelStationsIDBuf (§4.1)
Create quadtree for search area uv coordinates (§4.2)
Create acceleration structure for clash detection (§4.3)
$goalDensity \leftarrow$ ComputeGoalDensity($voxelSize$, $pointSpacing$) (§4.5.1)
{INIT Multiobjective OPTIMIZATION POPULATION}
for p = 1 to M do
  Init solution candidates coordinates randomly (§4.6)
  {nbCoordinates are $2N$ $(u, v)$ (FIX) or $2N_{max}$ $(u, v) + 1$ $(\alpha)$ (MIN) (§4.4)}
end for
{Multiobjective OPTIMIZATION}
nbIter $\leftarrow 0$
while (nbIter $<$ nbIterMax) do
  for p = 1 to M do
    $N_{stations} \leftarrow N$ (FIX) or $N_{min} + \alpha(N_{max} - N_{min})$ (MIN)
    voxelDensityBuf $\leftarrow 0$, voxelStationsIDBuf $\leftarrow 0$
    for i = 1 to $N_{stations}$ do
      if $u_i$ and $v_i$ found in triangle T of the search area triangle mesh (§4.2), then
        lidarPosition $\leftarrow$ UVtoXYZ(T, $u_i$, $v_i$ , $h$)
        $C_1 \leftarrow$ UVtoXYZ(T, $u_i$, $v_i$ , $h_1$), $C_2 \leftarrow$ UVtoXYZ(T, $u_i$, $v_i$ , $h_2$)
        if NOT Clash($C_1$, $C_2$, $r$, surveyed/blocking meshes) (§4.3), then
          pointCloud $\leftarrow$ SimulationLidar(lidarPosition, lidarParameters, surveyed/blocking meshes, meshes materials, computeNoise, noiseTable) (§3.2)
          voxelDensityBuf $\leftarrow$ ContribNbPoints(pointCloud, voxelKeysBuf) (§4.5.1)
          voxelStationsIDBuf $\leftarrow$ ContribStationID(pointCloud, voxelKeysBuf) (§4.5.2)
        end if
      end if
    end for
    {EVALUATE OBJECTIVE VALUES FOR CANDIDATE p}
    $coverage_p \leftarrow$ Coverage(voxelDensityBuf, $goalDensity$) (§4.5.1)
    $overlap_p \leftarrow$ Overlap(voxelStationsIDBuf) (§4.5.2)
    $nbStations_p \leftarrow \alpha$ (§4.5.3)
  end for
  UpdatePopulation(M, sigma) (§4.6)
  nbIter $\leftarrow$ nbIter + 1
end while
```

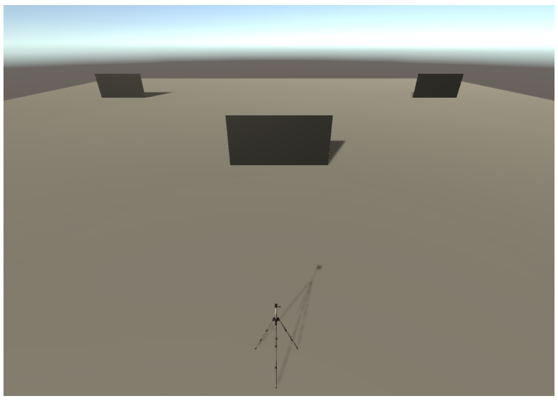
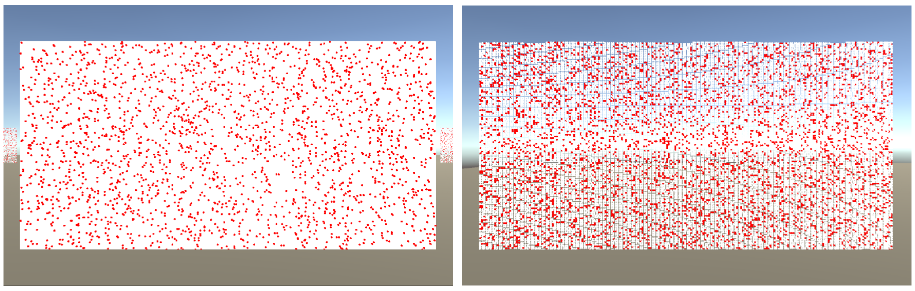


Figure 4: Scene with three walls, 5 m wide and 2.5 m high, placed at 10 m (center) and 30 m (sides) from a LiDAR station.



(a) Outliers on wall at 10 m.

(b) Outliers on wall at 30 m.

Figure 5: Impact of noise on coverage evaluation.

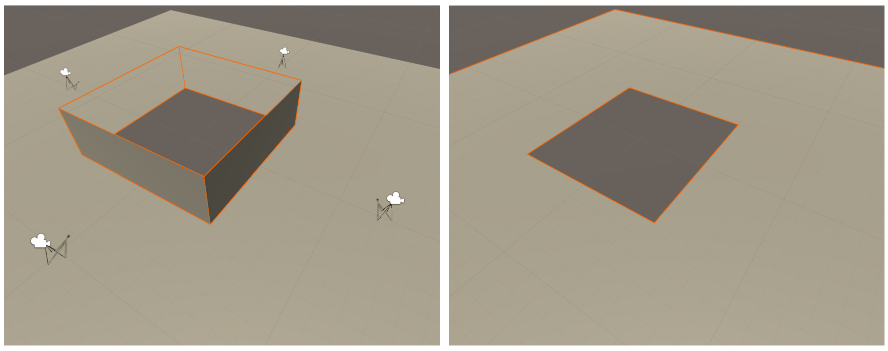
with grazing incident ray directions that generate small reflectivity values due to the Lambertian model used (see Section 3.3) – are penalized.

5.1.2 Four walls test case

Presentation of the test case In the second test, the scene contained four planar perpendicular walls placed on a surface such that they formed a hollow square (see Figure 6). The walls were the surveyed geometry and the floor was the search area. The surveyed mesh model contained 400 triangles.

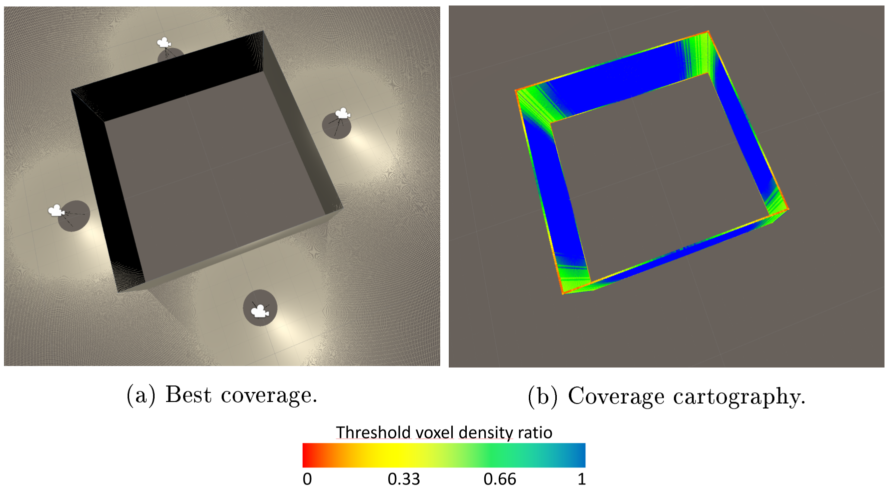
The user size was set to 10 cm, and the computed surveyed geometry voxel size was 7.8 cm. The point spacing was fixed at 1 point every 1 cm, which implied a goal density of 61.3 points per voxel. The LiDAR had a 360° horizontal wide angle, 300° vertical wide angle, and step angle of 0.1°, which corresponds to more than 5 million rays launched per station. The LiDAR height was 1.5 m. All simulations were performed without noise.

Coverage optimization for a fixed number of stations Using a COVERAGE_FIX optimization based on mono-objective CMA-ES for a network of four LiDAR stations, the best coverage reached after 200 iterations on a population of ten candidate solutions was 0.89. The same result was obtained regardless of the initial LiDAR position when performing several successive runs. The simulation and coverage estimation for each candidate required approximately 16 ms, for a total computation


(a) Scene with four walls, 10 m wide 3 m high, placed on a 40 m x 40 m floor with four TLS.

(b) Search area.

Figure 6: Simple four-wall scene.


(a) Best coverage.

(b) Coverage cartography.

Figure 7: COVERAGE_FIX optimization for four stations.

time[3] of 30 s. The resulting simulation for the best candidate solution is shown in Figure 7a, and its coverage cartography is displayed in Figure 7b.

This first result was not satisfactory, as the overlap for the optimal four-station configuration was 0, because the point clouds do not share any points on the surveyed geometry. Therefore, the question is how many stations are necessary and sufficient to reach almost complete coverage with more overlap.

Multiobjective optimization with a linear scalarization approach The method proposed in [4] was implemented with the following differences to allow fair comparisons:

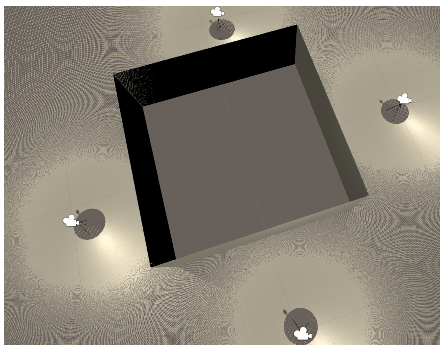
- Γ was evaluated using our density-based coverage estimation method (see Section 4.5.1).

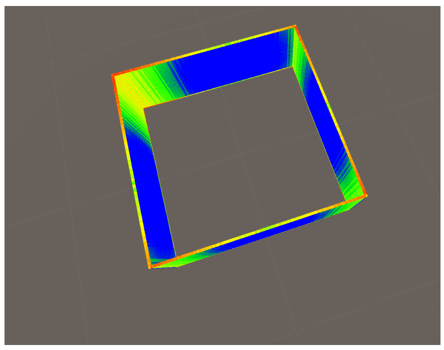
[3]In this study, all computations were performed on a Nvidia RTX 3090 GPU 24 GB RAM board.

- Φ, the function of the number of stations, was changed to $Phi = 2^{\frac{N_{max}-N_{stations}}{N_{max}-N_{min}}} - 1$ so that $Phi = 0$ for $N_{stations} = N_{max}$ and $Phi = 1$ for $N_{stations} = N_{min}$.
- The CMA-ES mono-objective method was used instead of the GA to maximize the fitness.

Otherwise, the function Δ allowing the measurement of LiDAR station graph visibility, as well as the weights λ_i, remained unchanged (see Section 2). N_{min} and N_{max} were fixed at 4 and 12 stations, respectively.

Using a population of 36 candidate solutions, the best solution found after 150 iterations performed in 75 s presented a fitness of 0.93. As shown in Figure 8, there was a slight variation in the solution obtained previously, with four stations moving away from the walls. Its coverage value Γ was 0.83, Φ was 1 as the candidate had $N_{min} = 4$ stations, and Δ was also 1 because of only one connected component as each station is within the visual range of at least an other one. The overlap value was still 0 because the simulated point clouds did not share any points.


(a) Candidate with best fitness.


(b) Coverage cartography.

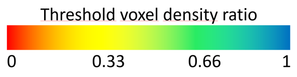


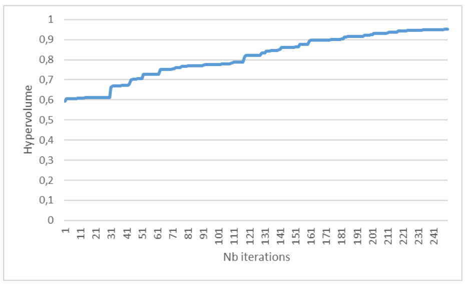
Figure 8: Candidate with best fitness found with a multiobjective linear scalarization approach.

Fitness maximization enables the maximization of Φ and Δ at the cost of a coverage value Γ lower than that obtained with our COVERAGE_FIX optimization (0.83 as opposed to 0.89). In fact, to increase Γ, the optimizer needs to either bring the stations closer to the walls, which lowers visibility and hence Δ, or to increase the number of stations, which lowers Φ. With the given weight values, both solution types are discarded, as they provide suboptimal fitness. If the user requires better coverage (almost perfect coverage, for instance), the has to increase the coverage weight λ_1 and decrease all other weights, thus indirectly setting his preferences.

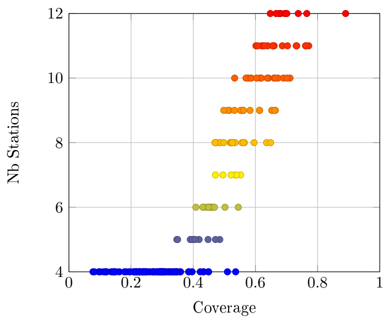
Our method is much more informative because it provides a Pareto front for solution candidates, which allows the user to easily interpret results and determine the best compromise. Finally, we believe that our overlap estimation objective, as formalized in Section 4.5.2, is more adequate than the proposed Δ function for measuring the registrability of LiDAR station point clouds.

Our multiobjective approach We proceeded in two steps. First, a COVERAGE_MIN multiobjective optimization was performed with $N_{min} = 4$ and $N_{max} = 12$. The population was set to 200 candidate solutions, and 250 iterations were performed for a total computational time of 928 s. The final hypervolume reached was 0.95

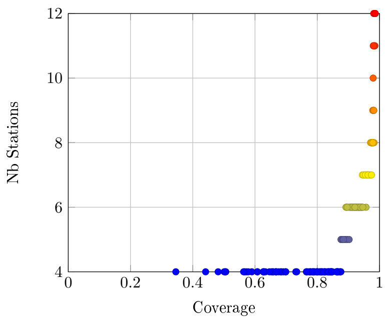
(with a maximum of 1, as shown in Section 4.6). Figure 9 illustrates the convergence of the Pareto front.



(a) Hypervolume.



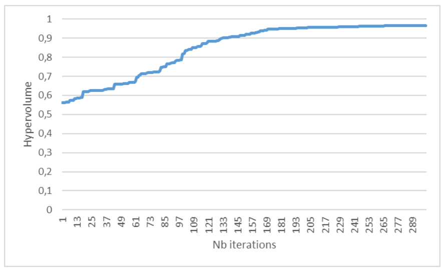
(b) Pareto front after 10 iterations.



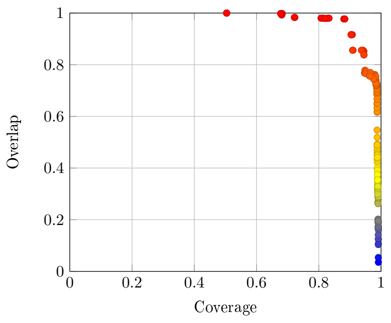
(c) Pareto front after 250 iterations.

Figure 9: COVERAGE_MIN for 4 to 12 stations.

As seen in Figure 9c, an increase in the number of stations beyond eight is not necessary because coverage of almost 1 is already reached. Indeed, the best candidate solution for eight stations reached a coverage of 0.982 and overlap of 0.26. Therefore, to find the best compromise on coverage and overlap, a second COVERAGE_OVERLAP_FIX multiobjective computation was launched with a fixed number of eight stations. The population size was again 200, and the number of iterations was 300. After computing for 1082 s, a hypervolume value of 0.97 was reached. Figure 10 shows both the hypervolume and final Pareto front.



(a) Hypervolume.



(b) After 300 iterations.

Figure 10: COVERAGE_OVERLAP_FIX for eight stations.

As can be seen in Figure 10b, the candidate that produced the best coverage, a point in blue on the lower-right, reached an almost perfect value of 0.99. However, because the station positions were too scattered, the overlap was only 0.16. By contrast, the candidate with the best overlap of 1.00, a point in red in the upper-middle part, reached a coverage of only 0.50 because the LiDAR positions were too grouped. A good compromise can be reached by choosing a candidate solution on the final Pareto front with coverage 0.98, slightly below the maximum, and an overlap of 0.75 (a reddish orange point in the upper-right part).

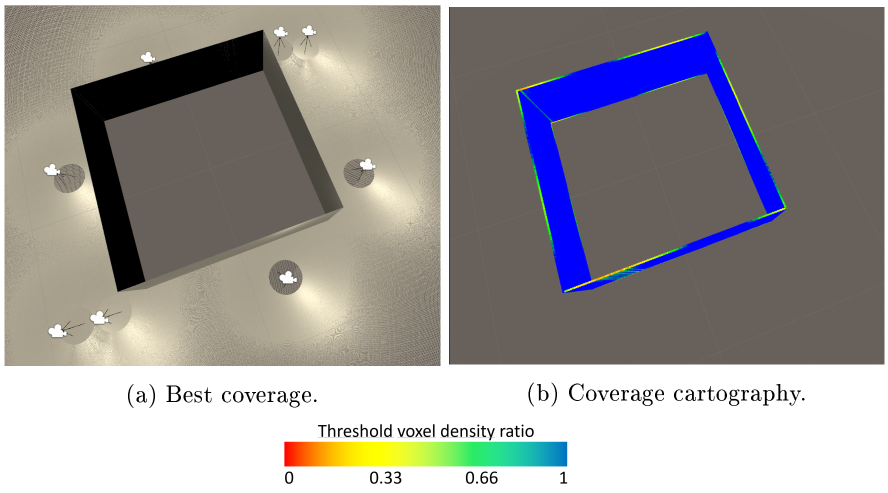


Figure 11: Best compromise for eight stations.

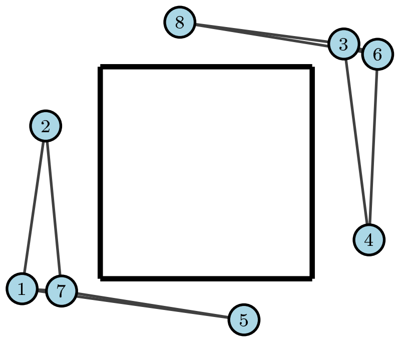


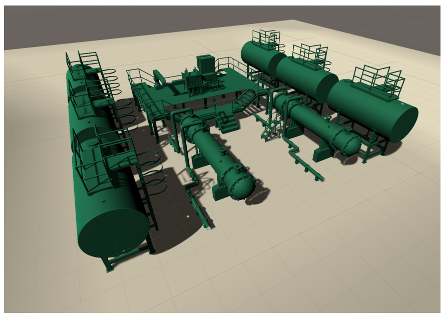
Figure 12: Best compromise overlap weighted graph.

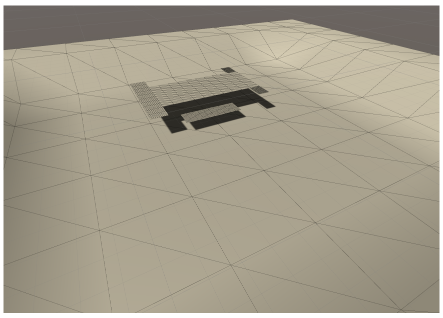
Figure 11 shows the LiDAR positions and cartography coverage of the candidate solution. Figure 12 shows the overlap weighted graph drawn around the four walls. Only connections with significant weights (≥ 0.01) are represented. All other connections are either strictly 0, as the point clouds do not share any points, or very small, as only points in the voxels of the wall edges are shared. The graph contains two connected components, each of which groups positions into triplets, which, as noted in Section 4.5.2, allows for more robust and precise global registration. The two components corresponding to the point clouds can be registered with the help of edge points shared between positions 2 and 8, and 4 and 5.

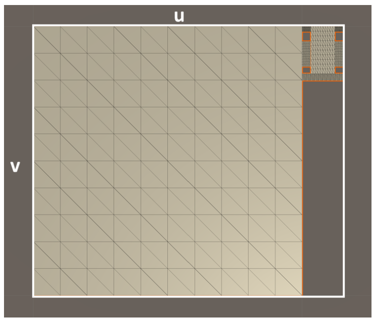
5.2 Complex case

. The 3D model shown in Fig. 13 represents a small industrial site that contains pipes, tanks, platforms, and relatively thin structures such as ladders, stairs, and railings. As in the original model, closed volumes such as tanks are thick, inner triangles were removed to avoid creating invisible voxels. The resulting model has 2.7 million triangles.

This surveyed model was placed on the ground with an embankment connected by a slope to its central flat part. The search area was composed of nonplanar ground together with different parts of the elevated platform model. Figures 13b and 13c illustrate this area in 3D and 2D (u, v) coordinates, respectively. It should be noted that in 2D parametric coordinates, the area of the ground is normalized, and the platform element areas have the same ratios relative to the ground as in 3D coordinates. Therefore, the (u, v) parametric coordinates encompass a rectangular area in the range $[0, 1.15]$x$[0, 1]$. The search area mesh contains 1400 triangles.


(a) Industrial site in green (15 m x 14.5 m x 4.5 m). The ground (40 m x 40 m) has an embankment 2.3 m high.


(b) Search area in 3D.



(c) Search area in 2D (u, v) coordinates.

Figure 13: Complex model.

The LiDAR simulation parameters are listed in Table 4.

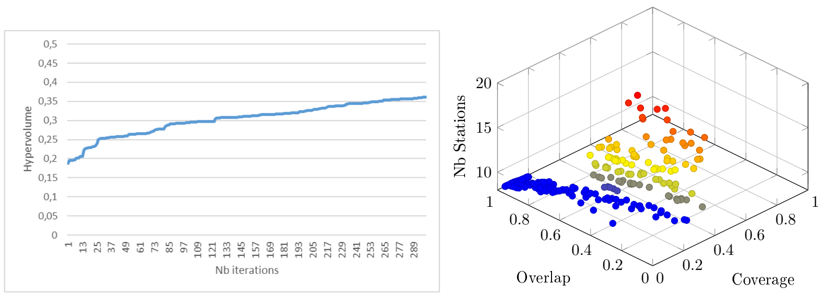
Point spacing was set to 1 point every 5 mm. The horizontal and vertical step angles allowed this spacing at a 10 m distance. More than 65 million rays were launched per station. The user size was 1 cm, and the voxel size was 0.78 cm. The goal density was 2.4 points per voxel. Noise was enabled for LiDAR simulations. Its parameters are listed in Table 1. The diffuse coefficient of the surveyed mesh material was assigned an arbitrary value of 0.24 in relation to the dark green aspect of the model.

Optimization search was performed in two steps:

Horizontal wide angle	360°
Vertical wide angle	300°
Step angles	0.0286°
Distance range $[d_{min}, d_{max}]$	0.1 - 130 m
LiDAR height h	1.85 m
Security volume h1	0.6 m
Security volume h2	1.35 m
Security volume r	0.5 m

Table 4: LiDAR simulation parameters.

- First, a broad phase search of type COVERAGE_OVERLAP_MIN with $N_{min} = 8$ and $N_{max} = 20$
- followed by a narrower phase search of type COVERAGE_OVERLAP_FIX for a minimal number of stations, allowing almost identical coverage to the previous N_{max}.

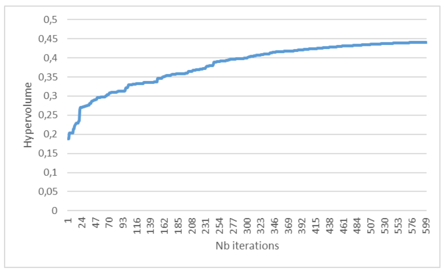


(a) Hypervolume. (b) Pareto front.

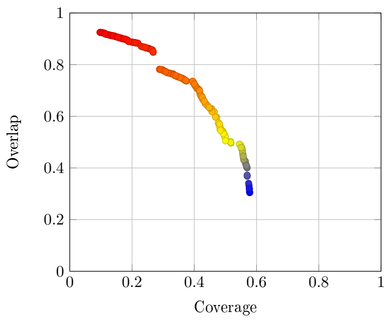
Figure 14: Complex model COVERAGE_OVERLAP_MIN.

For the first optimization step, 250 candidate solutions and 300 iterations were used. As shown in Figure 14a, the hypervolume curve is yet to reach its asymptote. Nevertheless, this provides us with the first useful insight. Each candidate evaluation, which includes LiDAR 3D position computation from (u, v) coordinates, clash detection around stations, LiDAR simulations, and objective value computation, requires an average of 314 ms. With a total of 75 000 candidates estimated, the overall computation time for this step was 23 421 s. The final Pareto front in 3D, with a hypervolume of 0.362, is presented in Fig. 14b. Here, the blue dots correspond to candidates with a small number of LiDAR positions. Although these points may have a high overlap of approximately 1, their coverage remains small, at 0.410 for the best eight positions candidate. By contrast, the red dots correspond to the highest coverage (0.542 for a 16-station candidate), while their overlap struggles to be sufficient. As the population no longer contains candidates with more than 19 stations (meaning they are dominated solutions), and considering the two best 15-station candidates rank 3rd and 5th with respective coverages of 0.520 and 0.501, while their overlap is 0.273 and 0.200, we decided to stick to 15 stations for the next step.

Subsequently, a second optimization search step was performed for 15 LiDAR stations, with a population of 250 candidate solutions and 600 iterations. The final Pareto front obtained in 2D (Figure 15b) reached a hypervolume of 0.441, which appeared near the asymptotic value (Figure 15a). The average computation time for a single candidate was 394 ms. Because 150 000 candidate solutions were evaluated



(a) Hypervolume.



(b) Pareto front.

Figure 15: Complex model COVERAGE_OVERLAP_FIX for 15 stations.

in total, the total optimization time was 59 106 s, i.e., more than 16 h. Again, a compromise was required. We chose to select the candidate with the sixth best coverage of 0.575. Although its coverage was slightly lower than the best case of 0.578, it produced a higher overlap of 0.340 compared with the 0.304 in the optimal coverage case. The 15 LiDAR positions and corresponding overlap-weighted graphs are shown in Figure 16. In Figure 16b, only links with weight values larger than 0.05 are displayed. The stations were closely connected, each with 1-6 significant overlapping links. A maximum $Overlap_{i,j}$ value of 0.37 was reached for Stations 2 and 9. Station 12 is on the platform, Stations 1, 5, 8, and 14 are on the slope, and Station 13 is on top of the embankment.

The global point cloud for this candidate solution with 15 stations reached 366.5 million points, the majority of which are on the ground. Amongst the 103.9 million intersection points with the surveyed meshes, 102.6 were found in the surveyed geometry voxels, which means there were very few outliers due to geometric noise. Figure 17 shows both the point cloud and its coverage cartography from different perspectives. The main area of the site was scanned with a satisfactory point density. Thin structures, such as ladders or railings, and small objects, such as bolts, can be clearly identified in the scan.

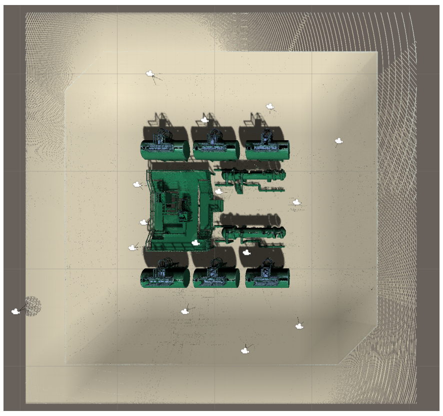
However, some parts could not be scanned correctly for the following two reasons.

- Some areas have limited access. This is the case, for instance, for a platform or between a tank and a main pipe (see Figures 18a and 18b, where the security volume is represented by two transparent orange spheres). When clash detection is performed, these areas correspond to very restricted and isolated areas in the (u, v) parametric search space. Therefore, the optimization method has difficulties reaching and exploring these places further.
- As the LiDAR height was kept constant, high (see Figure 18c) and low (see Figure 18d) areas could not be seen.

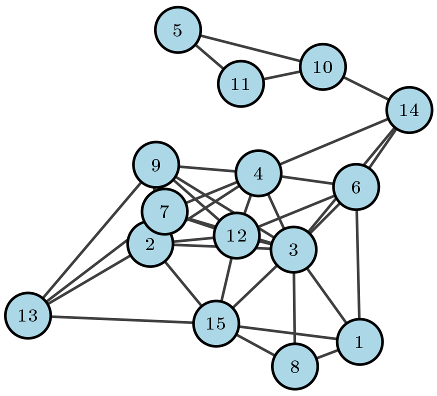
6 Possible improvements

In our method, most LiDAR parameters were fixed for each candidate. Some of these parameters could also be part of the optimization process. For example, LiDAR height can vary between stations[4], thus reaching areas that would otherwise remain inaccessible, as discussed in Section 5.2. Another parameter that may vary between

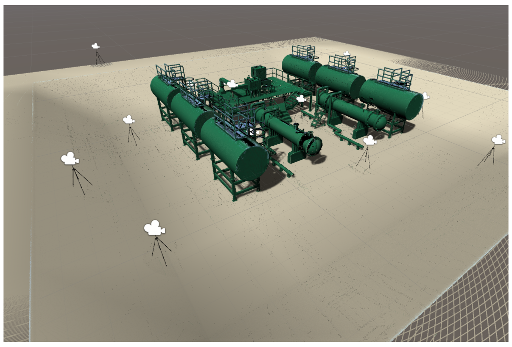
[4]In real world, the tripod legs can be shortened. Likewise, in order to reach high viewpoints, telescopic poles can be used.



(a) Top view of the 15 stations and simulated point clouds.



(b) Overlap weighted graph. Only the links with $Overlap_{i,j} > 0.05$ are shown.



(c) Perspective view.

Figure 16: Best positioning compromise found for 15 stations.

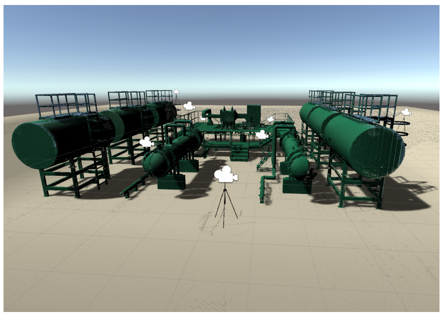
candidate solutions is the LiDAR step angle. Although these angles were fixed at the beginning of the survey, smaller step angles may provide solutions with fewer LiDAR stations.

Furthermore, in the studies presented in Section 5, the best solution was obtained after several iterative optimization steps. This workflow can be integrated into an interactive optimization process [38], where the user progressively chooses to focus the computational effort on parts of the search space.

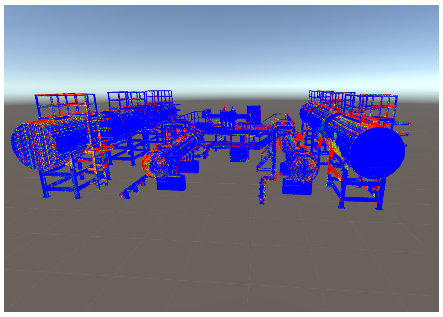
Finally, the LiDAR model could become more complex to enable more realistic scanning. This would enable a higher quality of noise and artifacts, such as mixed and isolated points, due to reflection and refraction. In survey cases where these phenomena are not negligible, this should provide an interesting improvement in the solutions.

7 Conclusion

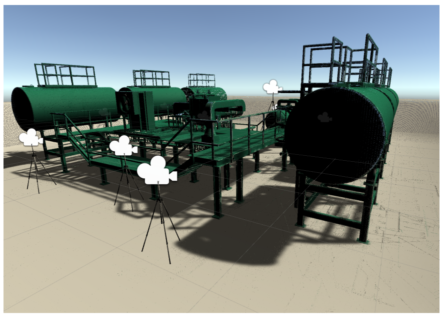
A novel method for optimizing the positioning of a network of terrestrial LiDAR stations on an a priori known 3D triangular mesh model was presented. This method relies on fast, GPU-based, and realistic 3D simulations that account for noise; explores complex search areas; and detects clashes using geometry. The use of a multiobjective approach allows for final decision-making based on rich and informative results.



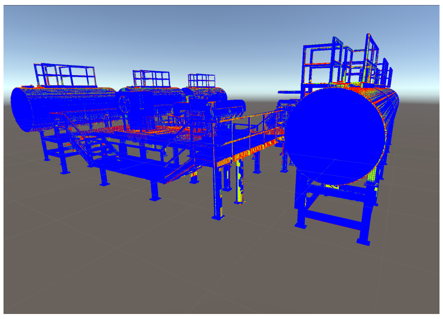
(a) General view.



(b) General view coverage.

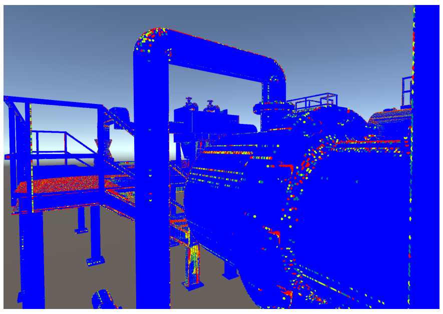


(c) General view (opposite side).



(d) General view coverage (opposite side).

(e) Close-up view.



(f) Close-up view coverage.

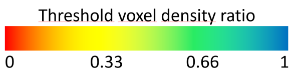
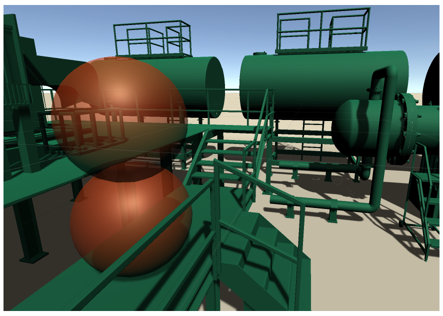


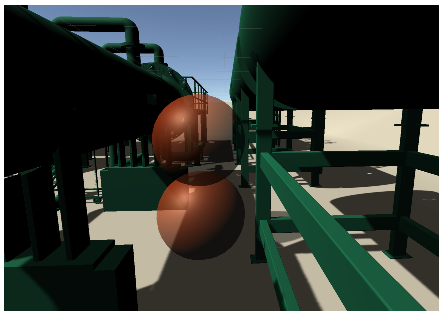
Figure 17: Simulated point clouds and coverage cartography for selected best candidate solution with 15 stations.

Experiments were presented on simple cases, and then on a complex industrial case, demonstrating the effectiveness of the method.

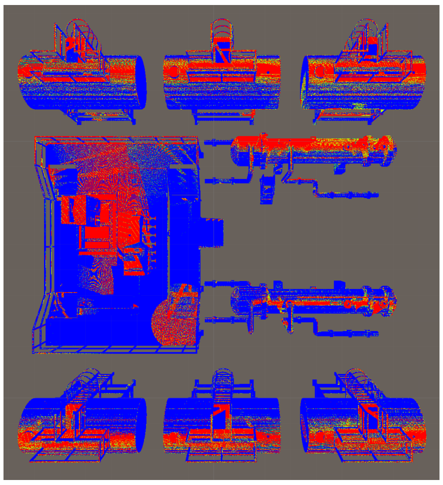
In the future, we plan to further accelerate computations using multiple GPUs to perform parallel evaluations of the candidate solutions for a given population at each iteration.



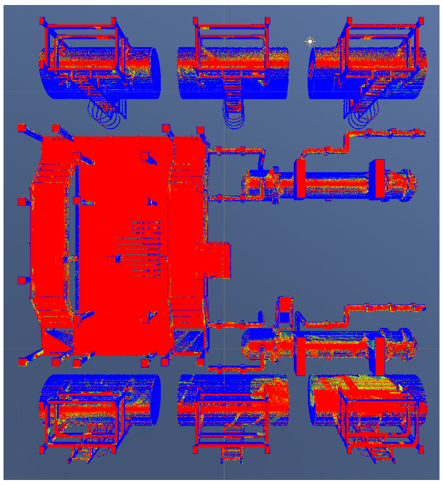
(a) Limited access to platform. Transparent orange spheres are the security volume.



(b) Limited access between pipe and tank.



(c) Coverage cartography - top view.



(d) Coverage cartography - bottom view.

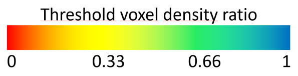


Figure 18: Difficult areas to scan.

Acknowledgements

3D models were obtained from the GrabCad website community library. We credit Alexey Epov for the factory structure model (Section 3.2), and Enes Hodzic for the industrial equipment model (Section 5.2).

We would like to express our sincere thanks to the anonymous reviewers, whose comments and suggestions helped improve and clarify this manuscript.

References

[1] Jaehong Ahn and Kwangyun Wohn. Interactive scan planning for heritage recording. *Multimedia Tools Appl.*, 75(7):3655–3675, apr 2016.

[2] Afrooz Aryan, Frédéric Bosché, and Pingbo Tang. Planning for terrestrial laser scanning in construction: A review. *Automation in Construction*, 125:103551, 2021.

[3] Emmanuel Benazera and Nikolaus Hansen. https://github.com/cma-es/libcmaes, 2014.

[4] Elena Cabrera Revuelta, María-José Chávez, José Antonio Barrera Vera, Yago Fernández Rodríguez, and Manuel Caballero Sánchez. Optimization of laser scanner positioning networks for architectural surveys through the design of genetic algorithms. *Measurement*, 174:108898, 2021.

[5] Meida Chen, Eyuphan Koc, Zhuoya Shi, and Lucio Soibelman. Proactive 2d model-based scan planning for existing buildings. *Automation in Construction*, 93:165–177, 2018.

[6] Youness Dehbi, Johannes Leonhardt, Johannes Oehrlein, and Jan-Henrik Haunert. Optimal scan planning with enforced network connectivity for the acquisition of three-dimensional indoor models. *ISPRS Journal of Photogrammetry and Remote Sensing*, 180:103–116, 2021.

[7] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. Carla: An open urban driving simulator, 2017.

[8] Michael T. Emmerich and André H. Deutz. A tutorial on multiobjective optimization: Fundamentals and evolutionary methods. *Natural Computing: An International Journal*, 17(3):585–609, sep 2018.

[9] M. Giorgini, S. Marini, R. Monica, and J. Aleotti. Sensor-based optimization of terrestrial laser scanning measurement setup on gpu. *IEEE Geoscience and Remote Sensing Letters*, 16(9):1452–1456, 2019.

[10] A. A. Goodenough and S. D. Brown. Dirsig5: Next-generation remote sensing data and image simulation framework. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 10(11):4818–4833, 2017.

[11] Michael Gschwandtner, Roland Kwitt, Andreas Uhl, and Wolfgang Pree. Blensor: Blender sensor simulation toolbox. In *Advances in Visual Computing*, pages 199–208. Springer Berlin Heidelberg, 2011.

[12] Yulan Guo, Hanyun Wang, Qingyong Hu, Hao Liu, Li Liu, and Mohammed Bennamoun. Deep learning for 3d point clouds: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(12):4338–4364, 2021.

[13] N. Hansen. The CMA evolution strategy: a comparing review. In J.A. Lozano, P. Larranaga, I. Inza, and E. Bengoetxea, editors, *Towards a new evolutionary computation. Advances on estimation of distribution algorithms*, pages 75–102. Springer, 2006.

[14] Nikolaus Hansen, Anne Auger, Raymond Ros, Steffen Finck, and Petr Posik. Comparing Results of 31 Algorithms from the Black-Box Optimization Benchmarking BBOB-2009. In *ACM-GECCO Genetic and Evolutionary Computation Conference*, Portland, United States, July 2010. pp. 1689-1696.

[15] Xiaoshui Huang, Guofeng Mei, Jian Zhang, and Rana Abbas. A comprehensive survey on point cloud registration, 2021.

[16] Christian Igel, Verena Heidrich-Meisner, and Tobias Glasmachers. Shark. *Journal of Machine Learning Research*, 9:993–996, 2008.

[17] Christian Igel, Thorsten Suttorp, and Nikolaus Hansen. Steady-state selection and efficient covariance matrix update in the multi-objective cma-es. In Shigeru Obayashi, Kalyanmoy Deb, Carlo Poloni, Tomoyuki Hiroyasu, and Tadahiko Murata, editors, *Evolutionary Multi-Criterion Optimization*, pages 171–185, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.

[18] Fengman Jia and Derek D. Lichti. A comparison of simulated annealing, genetic algorithm and particle swarm optimization in optimal first-order design of indoor tls networks. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, pages 75–82, 2017.

[19] Fengman Jia and Derek D. Lichti. A model-based design system for terrestrial laser scanning networks in complex sites. *Remote Sensing*, 11(15), 2019.

[20] Mi-Kyeong Kim, Bin Li, Je-Sung Park, Su-Jin Lee, and Hong-Gyoo Sohn. Optimal locations of terrestrial laser scanner for indoor mapping using genetic algorithm. In *The International Conference on Control, Automation and Information Sciences, ICCAIS 2014, Gwangju, South Korea, December 2-5, 2014*, pages 140–143. IEEE, 2014.

[21] Julien Kritter, Mathieu Brévilliers, Julien Lepagnot, and Lhassane Idoumghar. On the optimal placement of cameras for surveillance and the underlying set cover problem. *Applied Soft Computing*, 74:133–153, 2019.

[22] Christian Landgraf, Bernd Meese, Michael Pabst, Georg Martius, and Marco F. Huber. A reinforcement learning approach to view planning for automated inspection tasks. *Sensors*, 21(6), 2021.

[23] K. Majek and J. Bedkowski. Range sensors simulation using gpu ray tracing. In *Proceedings of the 9th International Conference on Computer Recognition Systems CORES*, 2015.

[24] S. Manivasagam, S. Wang, K. Wong, W. Zeng, M. Sazanovich, S. Tan, B. Yang, W. Ma, and R. Urtasun. Lidarsim: Realistic lidar simulation by leveraging the real world. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11164–11173, Los Alamitos, CA, USA, jun 2020. IEEE Computer Society.

[25] Morteza Heidari Mozaffar and Masood Varshosaz. Optimal placement of a terrestrial laser scanner with an emphasis on reducing occlusions. *Photogrammetric Record*, 31:374–393, 2016.

[26] Manon Peuzin-Jubert, Arnaud Polette, Dominique Nozais, Jean-Luc Mari, and Jean-Philippe Pernot. Survey on the View Planning Problem for Reverse Engineering and Automated Control Applications. *Computer-Aided Design*, 141:103094, December 2021.

[27] Guodong Rong, Byung Hyun Shin, Hadi Tabatabaee, Qiang Lu, Steve Lemke, Mārtiņš Možeiko, Eric Boise, Geehoon Uhm, Mark Gerow, Shalin Mehta, Eugene Agafonov, Tae Hyung Kim, Eric Sterner, Keunhae Ushiroda, Michael Reyes, Dmitry Zelenkovsky, and Seonman Kim. Lgsvl simulator: A high fidelity simulator for autonomous driving, 2020.

[28] Ahmet Saglam and Yiannis Papelis. Scalability of sensor simulation in ros-gazebo platform with and without using gpu. In *2020 Spring Simulation Conference (SpringSim)*, pages 1–11, 2020.

[29] Jari Saramäki, Mikko Kivelä, Jukka-Pekka Onnela, Kimmo Kaski, and János Kertész. Generalizations of the clustering coefficient to weighted complex networks. *Phys. Rev. E*, 75:027105, Feb 2007.

[30] Michael Schwarz and Hans-Peter Seidel. Fast parallel surface and solid voxelization on GPUs. *ACM Transactions on Graphics*, 29(6 (Proceedings of SIGGRAPH Asia 2010)):179:1–179:9, December 2010.

[31] Ming-Zhang Song, Zhenglai Shen, and Pingbo Tang. Data quality-oriented 3d laser scan planning. In *Construction Research Congress*, 2014.

[32] S. Soudarissanane and R. Lindenbergh. Optimizing Terrestrial Laser Scanning Measurement Set-Up. *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 3812:127–132, September 2011.

[33] Sylvie Soudarissanane, Roderik C. Lindenbergh, Massimo Menenti, and Peter J. G. Teunissen. Incidence angle influence on the quality of terrestrial laser scanning points. In *Proceedings ISPRS Workshop Laserscanning 2009, 1-2 Sept 2009, Paris, France*, 2009.

[34] Thomas Voß, Nikolaus Hansen, and Christian Igel. Improved Step Size Adaptation for the MO-CMA-ES. In *Genetic And Evolutionary Computation Conference*, pages 487–494, Portland, United States, July 2010. ACM.

[35] X. Wang, H. Zhang, and H. Gu. Solving optimal camera placement problems in iot using lh-rpso. *IEEE Access*, 8:40881–40891, 2020.

[36] Yu Wang, Eshwar Ghumare, Rik Vandenberghe, and Patrick Dupont. Comparison of different generalizations of clustering coefficient and local efficiency for weighted undirected graphs. *Neural Comput.*, 29(2):313–331, feb 2017.

[37] Lukas Winiwarter, Alberto Manuel Esmorís Pena, Hannah Weiser, Katharina Anders, Jorge Martínez Sanchez, Mark Searle, and Bernhard Höfle. Virtual laser scanning with helios++: A novel take on ray tracing-based simulation of topographic 3d laser scanning, 2021.

[38] Bin Xin, Lu Chen, Jie Chen, Hisao Ishibuchi, Kaoru Hirota, and Bo Liu. Interactive multiobjective optimization: A review of the state-of-the-art. *IEEE Access*, 6:41256–41279, 2018.

[39] Cheng Zhang, Vamsi Sai Kalasapudi, and Pingbo Tang. Rapid data quality oriented laser scan planning for dynamic construction environments. *Advanced Engineering Informatics*, 30(2):218–232, 2016.