Contents lists available at ScienceDirect

Automation in Construction

journal homepage: www.elsevier.com/locate/autcon

Metaheuristics for the optimization of Terrestrial LiDAR set-up

Alfonso López [a,*], Carlos J. Ogayar [a], Juan M. Jurado [b], Francisco R. Feito [a]

[a] Department of Computer Science, University of Jaén, Spain
[b] Department of Software Engineering, University of Granada, Spain

ARTICLE INFO

Keywords:
LiDAR
Metaheuristic
Genetic algorithm
GPGPU
Planning for Scanning

ABSTRACT

3D point clouds have a significant impact on a wide range of applications, although their acquisition is frequently conditioned by the occlusion of the objects in the scene. To address this problem, this paper describes an approach for optimizing LiDAR (Light Detection and Ranging) surveys using metaheuristics such as local searches and genetic algorithms. The method generates a set of optimal scanning locations to densely cover the real-world environment represented through 3D synthetic models. Compared to previous research, this paper handles 3D occlusion by varying the height of the sensor. Also, previously used metrics are compressed into three functions to avoid multi-objective optimization. Regarding performance, a LiDAR scanning solution based on GPU (Graphics Processing Unit) hardware is used. Several tests were conducted to show that the combination of local searches and genetic algorithms generates a reduced set of locations capable of optimizing the scanning of buildings.

1. Introduction

3D imaging technology is widely used in the construction industry for the tracking of building progress by enabling the acquisition of the environment geometry in a precise and highly detailed way. Instead of polygonal meshes, a discretized representation is given by point clouds. To achieve this goal, Terrestrial Laser Scanning (TLS) is increasingly being used to collect large sets of building data [1]. This technology can be applied to a wide range of applications, including building inspections [2], monitoring of natural environments (landform dynamics [3], ecological resilience [4], etc.), autonomous driving [5] and preservation of cultural heritage [6–8], among others. Besides terrestrial scanners, LiDAR (Light Detection and Ranging) technology presents multiple variants according to their capabilities (range, spatial resolution and covering, etc.) and the platform from which they are operated (Airborne (ALS), Backpack-mounted (BMLS), Mobile Mapping Systems (MMS), Terrestrial (TLS), Satellite (SLS), etc.) [9,10].

Some of the main challenges of TLS in the surveying of 3D facilities are the occlusion and range limitations [11]. Consequently, appropriate planning of TLS scans is necessary in order to (1) minimize the number of acquisition points, (2) generate a uniformly dense point cloud, and (3) reduce the occlusion from scene objects. These three objectives are equally influenced by the configuration and placement of the scanner. On the other hand, periodic scanning and monitoring of buildings are especially relevant for digitized representations, such as the widely known Building Information Modelling (BIM) [12]. They encode characteristics of a building, including 3D design drawings, materials, costs and safety specifications [13], and provide an interface for the management of 4D applications. Together with TLS, it allows the monitoring of continuously evolving buildings to preserve cultural heritage, track its current state and maintain repair records [7,8,14,15]. However, the monitoring of buildings over time is time-consuming, especially in dynamic environments. Also, TLS surveys generate multiple point clouds that need to be fused either by placing target marks [16] or by estimating the rigid transformation that minimizes the distance among overlapping point clouds. In order to speed up this acquisition task, the development of tools for the planning of TLS surveys plays a key role.

In the last few years, scanning on mobile platforms has arisen as an alternative to TLS. Mobile Laser Systems (MLS) reduce the acquisition time while still covering large areas [17]. However, the main drawbacks are the need of determining a path to appropriately survey the environment, the occlusion in complex environments as well as their low spatial density [18]. Moreover, operating a LiDAR from a mobile platform requires further optimizations to compute the optimal set-up regarding positioning [19].

We propose a methodology that solves the NP-complete problem of finding the best n-positions for TLS in 3D environments so that the scene coverage and the uniformity of the point cloud are optimized. This problem is assessed over 3D triangle meshes from BIM projects, consisting of one or multiple floors. For each one of these, multiple

* Corresponding author.
E-mail addresses: allopezr@ujaen.es (A. López), cogayar@ujaen.es (C.J. Ogayar), jjurado@ujaen.es (J.M. Jurado), ffeito@ujaen.es (F.R. Feito).

https://doi.org/10.1016/j.autcon.2022.104675
Received 13 April 2022; Received in revised form 14 November 2022; Accepted 15 November 2022
Available online 6 December 2022

positions (M) are uniformly sampled and enhanced using a spatial search according to an objective function. For that purpose, the neighbourhood is discretized and explored through minor translation vectors. Once the set M is improved (K), a Genetic Algorithm is used over K in order to find the best combination of n points, $N \subseteq K$, by evaluating several quality metrics. These metrics are computed from GPU-based LiDAR simulations described in previous work [20]. The latency of the overall methodology is reduced by taking advantage of GPU (Graphics Processing Unit) and multi-core CPU algorithms. Furthermore, modern data structures allow using highly detailed scenarios from BIM projects, and yet solve the optimization with low latency.

2. Related work

To the best of our knowledge, this is the first paper that investigates the optimization of LiDAR surveys within 3D environments by combining spatial searches, genetic algorithms (GA) and GPGPU computing (General Purpose Computing on Graphics Processing Unit).

2.1. Planning for scanning

The Planning for Scanning (P4S) has previously been addressed using a wide range of environments and techniques. If the number of target locations is known, then the selection of an optimum set is also known as the NP-complete set-coverage problem [11,21–23]. Previous research can be categorized according to the input environment, whether it is known (model-based) or not (non-model-based). The latter is mainly applied to robotic applications whose environment is unknown [24]. Otherwise, input scenarios are either defined as 2D or 3D models, with 2D representations being cross-sections of buildings [25]. These 2D-based solutions present lower computational complexity and are frequently solved following an iterative selection of viewpoints (Next Best View; NBV) or guided by heuristic algorithms [26]. The main drawback of these algorithms is that they do not consider the details underneath complex buildings. Instead, they are focused on 2D sketches composed of wall edges. 3D-based approaches are far more complex and guided by metrics that allow filtering and ordering space locations. Despite this, the exploration of 3D buildings is also approached with heuristics and NBV. However, the high latency frequently leads to scene simplifications, such as those based on voxelizations [27], Axis-Aligned Bounding Boxes (AABB) and Object-Oriented Bounding Boxes (OOBB) [28]. 2.5D models, such as Digital Surface Models (DSM), have also been investigated similarly to 2D environments [29].

2.2. Metaheuristics

Heuristics are the most frequent solver in P4S. Although they do not provide optimal solutions, they are proven good enough to cover environments with minimum scanning locations. Among heuristics, the Greedy algorithm has been extensively investigated [11,25,30,31], followed by Simulated Annealing (SA) [32,33], Genetic Algorithms (GA) [32,34], Particle Swarm Optimization [34] and Integer Programming [27]. Greedy algorithms are based on the iterative selection of locations, according to an objective function. Other Greedy variations weight the locations using a visibility score [34], are followed by SA [32], or optimized with Divide and Conquer (D&C) [30]. Instead of providing an automatic pipeline, Ahn and Wohn [35] proposed an interactive semi-automatic system.

Heuristic solvers are guided by objective functions measuring the quality of achieved solutions. To evaluate this, four metrics are frequently used in previous work [26,28]: Level of Detail (LOD), Level of Accuracy (LOA), Level of Overlap (LOO) and Level of Coverage (LOC). LOD refers to the point cloud resolution, LOA measures the quality of LiDAR returns, as higher distances and angles deteriorate the quality of the measurements [11,26], LOO refers to the overlapping area among point clouds so that TLS scans can be joined with rigid transformation estimations, e.g., Iterative Closest Point (ICP), and LOC refers to the number of polygons reached by scans. Thus, the optimization is constrained to limitations concerning range and angles.

Research on heuristics concerning 3D solutions is also frequent in the literature, though they are mostly linked to path planning [36] and the selection of subsets [36–38]. However, the number of scans for P4S to meet the required quality is uncertain in an infinite 3D space. The set of possible solutions is narrowed either by selecting random locations [33] or sampling the environment as a 2D grid [25,29,34]. For uniform subdivisions, the level of detail of the tessellation is a key factor with regard to response time. Coarse subdivisions present lower latency, though they are prone to yield far from optimal solutions. Starek et al. [29] enhances initial locations by applying minor translations while assessing their quality through an objective function, whereas Soudarissanane and Lindenbergh [11] improves the sampling by increasing the grid subdivisions, at the expense of higher response time. For 3D locations, Starek et al. [29] describes a simulating annealing procedure to transform uniformly sampled points into a surrounding location that improves the covering metric. Kim and Park [19] finds the optimum LiDAR position over an autonomous vehicle through a Genetic Algorithm (GA). For that purpose, this work utilizes the specifications of commercial LiDARs to compute the occupancy grid of sensor locations, defined as a discretized 360° map represented by several views acquiring the coverage region and dead zones. Beyond theoretical/simulation approaches, multiple studies focus on evaluating the set-up of several sensors, concerning height, angles and location in autonomous driving [39–41].

Once locations are locally optimized, they are processed as a classic set-coverage problem [11]. Recent research has solved this problem through GA approaches with different operators and configurations [22,23,42], though local searches [21] and other nature-inspired heuristics [38] are also reviewed. Despite heuristics being allowed to solve hard problems with optimal or nearly optimal solutions, they pose a challenge in terms of response time. Previous studies regarding P4S require days and hours to determine the optimal scan configuration for fine-grained grid subdivisions over 3D environments. Even 2D-based approaches suffer from high latency [25] whether they are implemented sequentially. Thus, multi-core and GPU-based algorithms offer a huge improvement in the response time, reducing it to a few seconds or minutes [25,37,42].

2.3. LiDAR simulation

Regarding previous work on LiDAR simulation, it covers a wide range of applications, from the design, validation, and calibration of LiDAR sensors [43,44] to the optimization of scanning processes [45–47]. Other studies focus on developing physically accurate LiDAR sensors by modelling multiple returns, beam scattering, surface properties or transmission medium [20,48–55]. Although LiDAR simulations are aimed at being realistic, they also involve stochastic features that are not relevant for finding optimal locations. Hence, this work evaluates the quality of a deterministic LiDAR by simplifying the simulation of López et al. [20]. Regarding efficiency, only a few studies are developed as high-performance solutions [20,56] capable of solving multiple dense simulations rapidly.

In conclusion, the main drawbacks observed in previous P4S work are (1) the widespread use of greedy algorithms to determine candidate scans, (2) the simplification of input scenarios to speed-up solutions, (3) objective functions based on non-smoothed thresholds (range, angle, etc.) and (4) the lack of parallel algorithms to solve both the simulation and the optimization. Consequently, the main contribution of this work is the planning of TLS surveys to reduce surface occlusion and generate dense point clouds by parallelizing the pipeline in the GPU. Objective functions are defined with smoothed boundaries, rather than thresholds, that score the fitness of individual scans, thus taking into account the LOA and LOD metrics. Also, candidate solutions are locally

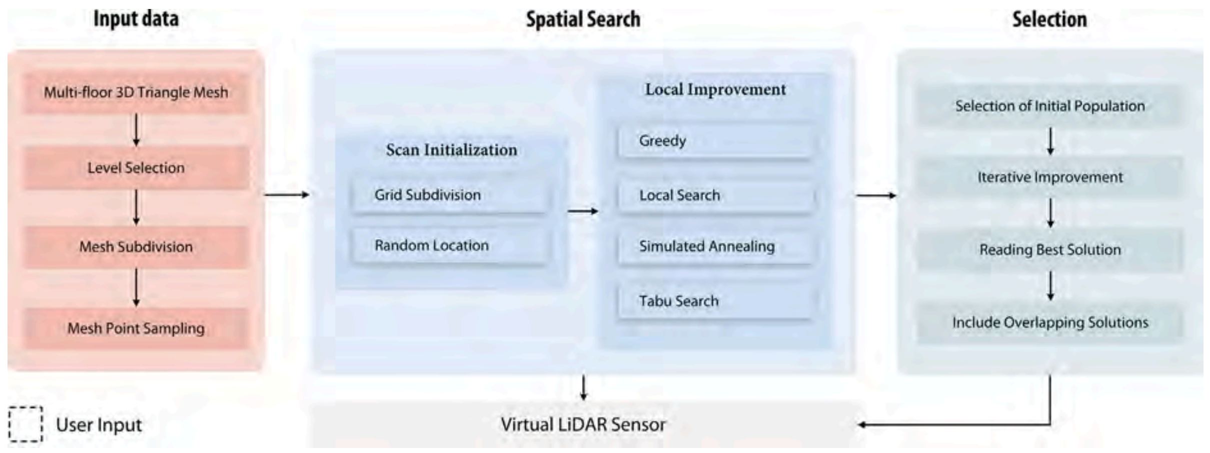

Fig. 1. Overview of the methodology of this work. The first stage is the pre-processing of input scenes from BIM projects. Then, a set of solutions is initialized and enhanced using a spatial search. Finally, a genetic algorithm is applied over the previous solutions to select the best-k that maximizes the proposed metrics. Finally, the selection is enhanced to guarantee the required overlap.

enhanced to avoid large space subdivisions. The optimal set of locations is computed using Genetic Algorithms, while implicitly minimizing the number of scans thanks to the objective function. The solution is further refined to guarantee the overlapping of individual scans using a Greedy approach. Our algorithm is evaluated with large CAD scenarios from BIM projects authored by Autodesk Revit®. As a result, this work is able to provide a near-optimal set of points for building monitoring efficiently. To this end, sensor configurations given by commercial devices are proven effective to perform the optimization, rather than defining fixed and non-intuitive thresholds.

This paper is structured as follows. The environment modelling is first described. Then, we present our optimization approach and implementation details. The results of the described methods are discussed in Section 4 to present the best configuration. Finally, the conclusions of this work are summarized in Section 5.

3. Material and methods

The details of the proposed algorithm are presented in this section, from input scenarios to the scan planning. The simulation within this work is based on previous work, and thus we refer the reader to López et al. [20] for further details. An overview of the proposed pipeline is shown in Fig. 1.

3.1. Environments

Input scenarios are triangle meshes composed of one or more floors and extracted from publicly available BIM projects. Scenes do not fit any standard and their geometry is initially unknown. As such, they are composed of polygons of varying size, with the largest triangles being part of the floor, walls and ceilings, whereas more dense geometry is found in furniture. The optimal scan configuration ought to cover as many polygons as possible. Hence, dealing with surfaces of similar size would greatly benefit the planning algorithm. To this end, the triangle mesh can be subdivided, despite perfect uniformity being not approachable due to GPU memory limitations and highly detailed items. Following this approach, triangles are recursively subdivided until their area is under a tolerance threshold.

Several approaches are effective to subdivide polygons, although some of them yield aesthetic results (see Fig. 2). Whether the edge to be split is selected randomly, the triangle can be subdivided into large triangles that degenerate into segments, thus causing missed collisions for a ray-casting LiDAR. Therefore, segment-like shapes along with precision error derived from the use of floating-point data may lead to the loss of LiDAR returns. Instead, we split the longest edge within each triangle, thereby generating triangles with uniform edge lengths.

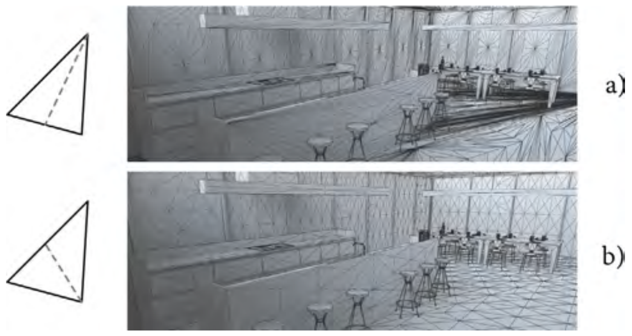

Fig. 2. Two different triangle subdivisions over the same scene. (a) Triangles subdivided by iterating through the edge to be split, and (b) triangles subdivided with the proposed method.

Regarding the indexing of input scenes, we build the Boundary Volume Hierarchy (BVH) using the GPU hardware to speed up spatial queries. Most of the previous research in LiDAR simulators solves the ray-casting problem in the image space through z-buffers, i.e., depth buffers. Similarly to colour representation in OpenGL's buffers, depth-buffers store the distance of the nearest object visible for each pixel using values in $[0, 255]$. With this approach, LiDAR simulations are massively parallelized but also lack precision. On the other hand, an efficient solution to the ray-casting challenge requires an optimized data structure, such as the BVH. It is a binary tree that is popular among ray-tracing applications since it allows discarding a significant number of polygons during the tree traversal. Thus, it copes with the management of large triangle meshes and notable amounts of cast rays.

Building a BVH as well as solving spatial queries present high latency when performed sequentially. To avoid this, we generated the BVH using the massively parallel method proposed by Meister and Bittner [57]. This work builds it on the GPU using OpenGL's compute shaders, by sorting primitives according to their Morton codes and merging them up to the tree root. As a result, triangle meshes of up to several millions of triangles are organized in a BVH with a latency in the magnitude of milliseconds. Finally, the BVH traversal is also accelerated by making several threads work in parallel to solve the collisions of LiDAR rays.

3.2. Solution encoding

Candidate solutions for genetic algorithms are encoded as a buffer of binary values indicating which minimal set of LiDAR positions

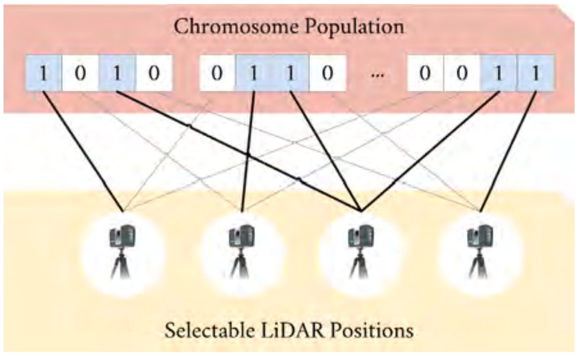

Fig. 3. Binary encoding of active LiDAR solutions for a genetic algorithm.

represents an optimal set-up to cover the scene. Each binary value is defined as an activation value for a LiDAR scan. Firstly, positions are uniformly or randomly sampled along the selected building floor. Then, they are optimized by measuring the effect in the metrics of small spatial moves. Once evaluated, the genetic algorithm is aimed at activating the minimum number of positions while maximizing the scenario coverage of a LiDAR point cloud.

Consequently, initial solutions are stored as x, y and z coordinates, while the solutions of the genetic algorithm are encoded as a vector of binary values $[b_0, b_1, b_2, \dots, b_{k-1}]$, with $b_i \in \{0, 1\}$ and k defined as the number of solutions generated by the previous spatial search. Despite the fact that activating k solutions provides the most complete scan, the genetic algorithm is expected to activate fewer locations. Fig. 3 shows the proposed solution encoding. Although binary values can be represented by Boolean values in the CPU, this data type is not uniformly represented in CPU and GPU hardware concerning data size, thus hardening data transfers. In our solution, Boolean values are expressed through the minimal integer encoding in the GPU hardware, given by the `uint8_t` data type from GLSL's `GL_NV_gpu_shader5` extension (OpenGL Shading Language). Representations with a lower GPU memory footprint are indeed possible through bit-level encoding, at expense of more intricate operations for index access. Moreover, the scenarios evaluated in Section Results and discussion achieve only a few MBs in the worst case.

3.3. Metrics

The goal of a fitness function is to evaluate the quality of candidate solutions and allow the optimization method to improve solutions and discern which are the best for a specific configuration.

3.3.1. Level of accuracy, coverage and resolution

Solutions to this optimization problem are given by n different positions within a scene level. Three metrics are considered to evaluate the fitness of a solution. First, the number of scanned polygons provides a raw measure regarding the coverage of the scene geometry (F_1; LOC). Also, polygons should be scanned with multiple points uniformly distributed (F_2; LOD, LOA, LOC). Finally, TLS scans should reach a minimum overlap factor that guarantees they can be joined during post-processing (F_3; LOO). The first metric is easily solved by counting different intersected polygons. On the other hand, the uniformity measurement of a point cloud requires more intricate algorithms. Instead of providing the aimed resolution as a parameter, it is automatically computed as part of pre-processing. Thus, polygons are assigned a nearly optimal average distance between uniform points. To this end, we consider the parametric equation of a triangle and randomly sample u and v variables with a uniform distribution. Therefore, setups whose mean distance is higher than the reference mean distance for a polygon are evaluated with a worse fitness value. As a result, point clouds that cover a polygon with a few points are heavily penalized (Fig. 4). The QMC (Quasi-Monte Carlo) sampler approximates better what is expected as the output of a LiDAR scan while the average point distance is similar to the uniform grid sampling.

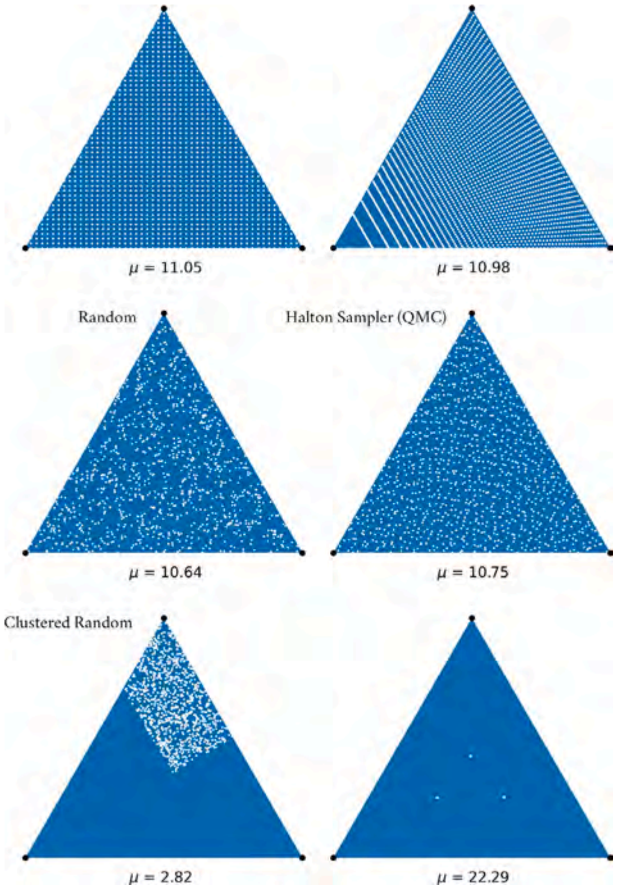

Fig. 4. Average distance (μ) of every sampled point with the rest of the points. From left to right, and from top to bottom: uniform grid distribution, uniform random sampling of u and v, random distribution, QMC random distribution, random distribution only for a part of the triangle and three points uniformly scattered. Note that the QMC distribution is used as the reference for our F_2 metric.

The described fitness metrics allow enhancing a given position, thereby orienting it towards better locations. Despite being described for a single scan, it can be applied to several of them. Nevertheless, point cloud uniformity and accuracy are preferred over coverage to enhance a single location. On the other hand, the selection of a subset of positions is better guided by the surface coverage. Despite this, both metrics are considered during optimization, with secondary fitness functions working as untying operators. Therefore, both metrics are defined as follows whether we aim to evaluate a set of solutions, K (Eqs. (1) and (2)). Individual solutions present a buffer of collided triangles, T, where each triangle is reached by a set of points (P). The set of unique triangle indices which were collided is here denoted by L, whereas the whole set of scene triangles is S.

$$F_1 = \sum_{s=1}^{|S|} 1[t_s \in \{L_1, L_2, \dots, L_k\}] \tag{1}$$

$$F_2 = \sum_{k=1}^{|K|} \left(\sum_{s=1}^{|S|} d_{optimal_s} - \sum_{l=1}^{|L_k|} d_{optimal_l} + \right.$$

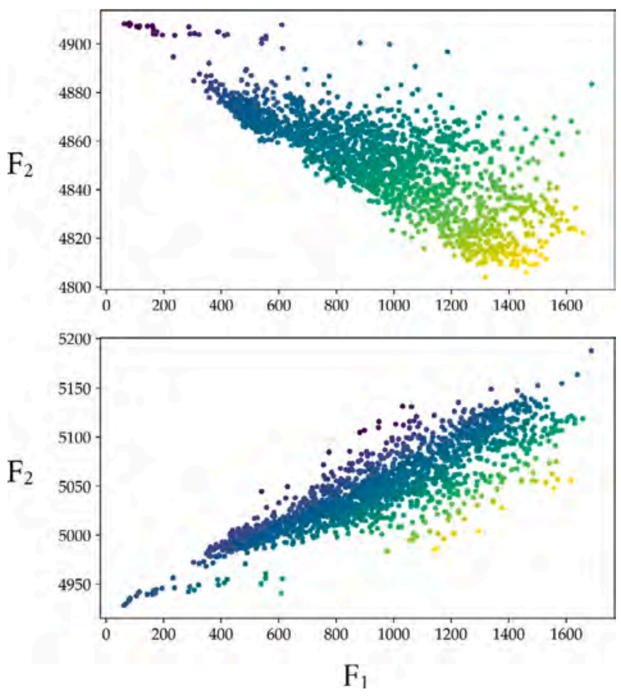

Fig. 5. F_1 and F_2 results obtained by uniformly sampling a 3D environment. The first image depicts the results of F_2 as proposed. The second image omits the second sum term, i.e., does not remove the default distance of collided polygons.

$$+\sum_{t=1}^{|T_k|}\left(\frac{\sum_{i=1}^{|P_t|}\frac{\sum_{j=1}^{|P_t|}d^2(p_i,p_j)}{max(|P_t|-1,1)}\cdot(2-|\hat{n}_{t_k}\cdot\widehat{(r_o-p_i)}|)}{max\left(|P_t|,1\right)}-\right.$$

$$\left.-d_{optimal_t}\right)\Bigg)\Bigg) \quad (2)$$

where F_1 yields the number of unique intersected polygons and F_2 represents the accuracy metric by measuring the distance from the expected uniformity to the obtained value. d represents a distance function, naively given by the Euclidean distance, t_s is a triangle index on the set S, $\hat{n}$ is the normal vector of a triangle, and r_o is the LiDAR location, i.e., the ray's origin. Instead of only accounting for the distance from intersected polygons, it also sums the distance from non-collided polygons. However, to avoid penalizing solutions with a higher number of collisions in a minimization problem, the distance from intersected polygons is subtracted (Fig. 5). Hence, the number of reached polygons is also integrated into this formula to penalize scans that minimize the foundational accuracy terms by reaching a small number of polygons.

Also, note that a frequently integrated criterion in the LOA metric is the sensor's range. Accordingly, previous work has included range in the LOA definition by omitting returns whose distance is above a threshold. However, these criteria can be integrated into the LiDAR simulation rather than into the above formula. Therefore, the range of our LiDAR simulation is considerably reduced to solely consider close collisions, although real simulations may provide denser scans.

3.3.2. Level of overlap

F_2 metric is better suited for performing local enhancements, whereas F_1's objective is to select the minimum set of locations that provide better scene coverage. However, none of these metrics considers the overlapping of individual scans, despite it being a required feature to guarantee their alignment in post-processing. To this end, features visible in a single scan ought to appear in more than one scan. With this objective, the F_3 metric defined in Eq. (3) measures the percentage of overlapping area between two LiDAR scans, i and j, depicted as circumferences with a fixed radius.

$$A_{i,j}=\left(r_i^2\cos^{-1}(\frac{d_i}{r_i})-d_i\sqrt{r_i^2-d_i^2}+\right.$$
$$\left.+r_j^2\cos^{-1}(\frac{d_j}{r_j})-d_j\sqrt{r_j^2-d_j^2}\right)$$
$$A_{i,j,r_i=r_j}=2\left(r^2\cos^{-1}(\frac{d}{2r})-\frac{d}{2}\sqrt{r^2-\left(\frac{d}{2}\right)^2}\right)$$
$$F_{3_{i,j}}=\frac{A_{i,j}}{\pi r_i^2}\ \mid\ i,j=1,2,\ldots,|P_c| \quad (3)$$

with $d=d_i+d_j$, where d is the distance from two sensor placements, and r is the LiDAR maximum range, which is previously clamped to account for the accuracy loss from distance. Due to r being the same for every scan, $d_i=d_j$.

The main drawback of Eq. (3) is that it does not account for the overlapping of previously added locations. To handle this, the area of influence of each LiDAR scan is represented by a 2D grid to be filled by points sampled from other scans. Hence, an approximated overlap is given by the number of voxels filled with respect to the overall number. Despite this, sampling can be avoided whether $d\geq d_1+d_2$. Yet, Eq. (3) is useful for establishing a ranking of candidate solutions that can help to achieve the required overlapping. However, ranking locations by their overlapping leads to selecting the highest overlap possible, instead of the required one. Accordingly, Eq. (4) ranks locations by their distance to the required overlap. However, solutions that offer a higher overlap than the required one are preferred over those below. To guarantee this, the most significant bit of preferred locations is set to one. Due to the described sampling strategy, the accuracy of the measured scan overlapping depends on the grid subdivision and the number of sampled points belonging to another TLS scan, as depicted in Fig. 6.

$$F_{3_{i,j}opt.}=r-|o-F_{3_{i,j}}|$$
$$F_{3_{i,j}opt.}=F_{3_{i,j}opt.}\mid 1\ll 32\quad if\ F_{3_{i,j}}\geq o \quad (4)$$

with o being the required overlapping percentage.

With this approach, the required overlapping is guaranteed for every solution. However, disjoint sets can be found, especially as a result of greedy algorithms selecting the best n solutions. To this end, the disjoint set is built by linking overlapping LiDAR solutions. Hence, new solutions are added until there is a single disjoint set, according to their distance to another disjoint set and the value of F_3, as proposed in Eq. (5). The procedure to join disjoint sets is implemented as follows:

- First, the closest disjoint sets are selected, as well as the two closest solutions of both of them.
- Then, solutions overlapped with the first one are sorted according to a maximization problem guided by Eq. (5).
- The new LiDAR scan is linked to the first solution, which is known to be overlapped, whereas the second solution is only liked whether the overlapping is higher than zero.
- Newly included solutions are also checked to guarantee the required overlapping.

$$g(p_o,p_d,p_i)=\left(\frac{d(p_o,p_d)-d(p_i,p_d)}{d(p_o,p_d)}F_{3_{o,i}opt.}\right) \quad (5)$$

3.3.3. Point sampling

This procedure is a preprocessing stage, although it can be performed later to re-sample the polygons. In this stage, the polygons are iteratively processed to generate uniformly distributed points. Then, the

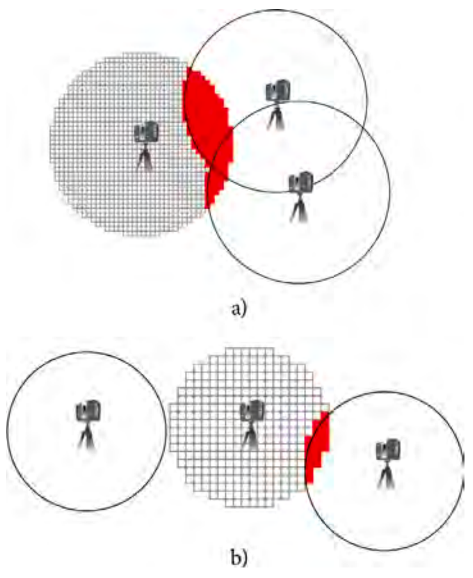

Fig. 6. Grid occupancy in two different configurations. (a) Grid with higher resolution and three overlapping circumferences, and (b) sparser grid with only two overlapping circumferences.

mean distance is computed as shown in Eq. (6) for a single triangle whether we apply the Euclidean distance. Note that $|P|-1$ is used instead of $|P|$ to omit taking into account the distance of points with themselves.

$$d_{mean} = \frac{\sum_{i=1}^{|P_t|} \frac{\sum_{j=1}^{|P_t|} d^2(p_i - p_j)}{max(|P|-1, 1)}}{max\,(|P|, 1)} \tag{6}$$

where P_t represents a set of points within a triangle t and the inner summation computes the Euclidean distance for xyz points.

Despite there exist built-in random uniform distributions in most language libraries, these are mainly based on pseudo-random sequences, such as those used for Classic Monte Carlo integration (CMC). They provide good quality estimations, though they require a large number of samples and thus present a higher response time [58]. In our case study, the response time is irrelevant as we solely sample a reduced number of points that depends on the polygon area. However, it generates a point cloud far from the convergence scenario, i.e., uniformly sampled points. Instead, the QMC method is based on well-distributed deterministic sampling patterns that provide better results for our objective. More specifically, we apply the Halton sequence to calculate the quasi-optimal mean distance within polygons [59,60]. Fig. 7 shows the results of sampling with both the C++ built-in random uniform distribution and the proposed QMC sequence.

Either from pseudo-random or QMC sequences, two parametric values (u, v) are generated to sample a triangle surface. Eq. (7) shows the formula for generating a point within a triangle defined through its three vertices (p_1, p_2, p_3) with $u, v \in [0, 1]$.

$$p_s = (1-\sqrt{u})p_1 + (\sqrt{u}(1-v))p_2 + (v\sqrt{u})p_3 \tag{7}$$

The main drawback of this solution is that sparse scans, with points gathered in a small area, would provide a good fitness result. To avoid this, the triangle's vertices are included as part of the point set, $|P|$, thus penalizing scans biased towards small parts of polygons.

3.4. Solution initialization

The continuous 3D space must be discretized to tackle the P4S. The selection of N initial solutions depends on polygons interactively marked as ground, as regarded in Section Level selection. From the collected ground planes, we compute the 2D axis-aligned bounding box (AABB) to restrict the instancing area. However, this solution may generate points out of ground-labelled polygons for non-rectangular ground planes. Therefore, candidate locations are evaluated on the GPU to verify the following conditions:

1. **The sensor is located over a ground polygon**. To verify this, we cast a ray towards $-Y$ using the described BVH data structure to speed up queries. Ground polygons are transferred to the GPU and their ID is sequentially compared to the nearest found collision.
2. **The sensor is not located over non-ground planes**, e.g., objects placed between the floor and LiDAR's y coordinate. m points are sampled around the candidate location, using a radius r and casting m rays with $-\vec{Y}$ direction. It is discarded whether any of the m rays collide with non-ground-labelled items. Both r and m can be configured.
3. **The sensor is not located close to building walls and furniture**. It cannot be placed over items, nor next to them, thus guaranteeing a safety distance (d). Hence, four additional rays are cast using $\{X, -X, Z, -Z\}$ vectors.

Candidate solutions can be generated through random or uniform sampling. Uniform sampling uses a 3D matrix bounded by the ground's 2D AABB. Height is limited by LiDAR's maximum and minimum height. The matrix subdivision is parameterized by the voxel size. Otherwise, we can randomly distribute solutions instead of intensively sampling the 3D space. The objective of random sampling is to explore fewer solutions, though scattered throughout the scene. Hence, a random uniform distribution is used instead of a QMC sampler. This initial set of solutions can be either enhanced (Spatial Search), subsampled (Greedy approach) or selected through bit sets (Genetic Algorithm).

3.5. Spatial search

The following section describes how the initial solutions are improved, thus providing a wider space exploration, instead of relying on initial locations. However, metrics for improving individual scans do not integrate any knowledge of overlapping among scans. The following searches work according to the F_2 metric defined in Eq. (2).

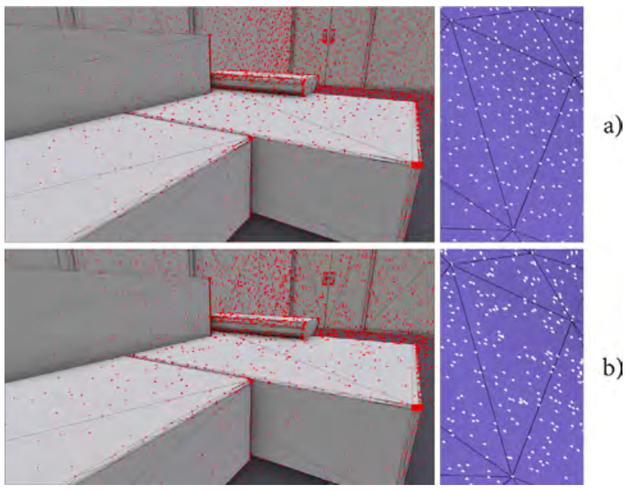

Fig. 7. Comparison of polygons sampled with different sequences. (a) The point cloud obtained using the Halton sequence is significantly more uniformly sparsed than (b), sampled with a random uniform distribution.

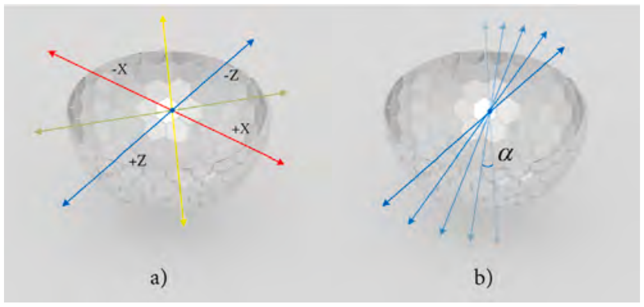

Fig. 8. Overview of gradients for a 2D neighbour search. (a) Constant approach that generates 8 vectors, whereas (b) sub-samples the circumference according to a resolution parameter.

3.5.1. Solution neighbourhood

Neighbourhood exploration is here introduced to locally enhance candidate locations. The neighbourhood is presented as 3D points close to the currently evaluated position. It poses several challenges regarding discretization, as it initially consists of an infinite set of translations. Firstly, the neighbourhood size can be significantly downscaled by selecting 14 unit vectors, given by the faces (6) and corners (8) of a unit cube. Otherwise, the neighbourhood can be discretized by uniformly subsampling a unit sphere, thus obtaining a buffer of gradients whose length depends on the user-defined sampling resolution. Furthermore, unit vectors can be scaled during the spatial search by defining the step length. Note that a larger length leads to re-exploring parts of the scene, whereas an extremely small length may not provide any enhancement to the current solution. Fig. 8 illustrates the 2D gradient vectors to explore the contiguous space of a point located at the sphere centre. In 2D, 8 different gradients are generated in the first scenario, whereas the number of gradients in the second depends on the resolution.

3.5.2. Greedy local search

The Local Search (LS) algorithm is implemented to assess if the space can be rapidly surveyed. For that purpose, the neighbourhood of each solution is explored, evaluated and ordered according to its fitness. Solutions are iteratively improved by moving to the best solution in the neighbourhood if any improves the current fitness. Otherwise, LS falls on a local minimum and terminates the process. It also happens whether the iteration exceeds a threshold, avoiding loops. However, immediate loops as a result of revisiting previously explored solutions can be efficiently discarded. To this end, we compute the dot product of the gradient to be assessed and the last gradient ($f(\hat{g}_1, \hat{g}_2) = \hat{g}_1 \cdot \hat{g}_2$). A gradient is considered to be already explored whether $f(\hat{g}_1, \hat{g}_2) < \epsilon - 1$, with ϵ being a constant close to zero, in order to deal with floating-point errors.

3.5.3. Simulated annealing

Simulated Annealing (SA) is evaluated as it tackles getting stuck in local optima. To this aim, neighbours with worse fitness than the current solution can be accepted following probabilistic criteria. A temperature value (T) is first initialized and reduced iteratively. The acceptance rate of worse solutions is lower as T decreases (cooling) [34, 61]. In this work, it is configured using the traditional exponential formula, as defined in Eq. (8) for a minimization problem:

$$p(f') = \begin{cases} \exp(\frac{f'-f}{T}) & f' - f \geq 0 \\ 1 & f' - f < 0 \end{cases} \tag{8}$$

given that f' and f are two fitness values from a new solution and the current one, respectively. Consequently, a random uniform distribution is also applied to determine whether a worse solution is accepted. The stop criterion depends on a threshold temperature and a maximum number of iterations.

3.5.4. Tabu search

The improvement of a naive local search leads to the Tabu Search (TS) algorithm, which saves the so-called tabu moves that were already processed and cannot be explored again. Re-initialization is performed whether all neighbours are tabu moves, their objective value is worse than both the current solution and overall best solution, or a maximum number of iterations without improvement has been achieved. In 3D, the main challenge is to handle the tabu move list in a continuous space. Therefore, we narrowed the 3D space using a regular grid, i.e., a 3D matrix with variable resolution. A move is considered tabu when the target point falls in a tabu voxel. Given the step size of neighbour explorations, n_l, an appropriate voxel size is given by $f \cdot n_l$, with $1 < f < 2$.

3.6. Genetic algorithm

Genetic Algorithms (GA) are aimed at selecting a subset of previous locations. For that purpose, this sort of algorithm is inspired by nature to handle populations that evolve and improve over time. Accordingly, this method combines both exploration and exploitation phases [62]. Regarding our case study, this metaheuristic helps to reduce the setup complexity while still providing a dense coverage. In comparison with memetic algorithms, the first stage aims to improve individual locations guided by an exploration of surrounding areas, without evaluating the scene coverage. Nevertheless, the proposed F_2 metric takes into account the weight of non-scanned polygons. Then, the final GA selection tries to find the optimal solution regarding the scene coverage. To this end, a slight variation of the previously proposed F_1 metric is here proposed. The complete procedure is depicted in Fig. 9, whose stages are following detailed:

Initialization of population. The population is first generated as p chromosomes of size n with a random number of activated LiDAR locations. Hence, variety is ensured by avoiding populations with a fixed number of activated bits.

Compute template rays. Template rays are first built to describe how rays traverse the space from a default LiDAR location, which is known to be $(0, 0, 0)$. These rays are generated according to the LiDAR configuration, which ought to follow the specifications of a standard model. The LiDAR simulation receives as input a set of rays which are following solved to return collisions. Hence, template rays are replicated in the GPU for each location to be tested, thereby solving them all at once. Otherwise, a single LiDAR simulation ought to be performed, once for every location. However, the GPU-based LiDAR simulation from previous work is simplified to provide only the first collision both to favour replicability and performance of experiments. As a result, the template rays are built once in the GPU and used multiple times with different locations.

Parent selection. The aim of this stage is to select the s most promising individuals for the subsequent crossover. Two different selections are here proposed. First, the 2-by-2 tournament allows selecting good solutions, though they may not be within the top-most s. Therefore, this leads to introducing some minor variety into later populations. On the other hand, the elitist approach selects the best individuals from the current population.

Crossover. Previously selected parents are here combined to generate the next population. Parents can be mixed by selecting a crossing point, thus generating two gen chunks for each parent, which are combined to build offspring. Gens can also be individually processed by retrieving uniformly distributed random values which determine the parent (p_{1_i} if $r_i \leq 0.5$, p_{2_i} otherwise). There is only a non-valid solution given by a zero chromosome, which is avoided by randomly altering a bit.

Mutation. Some minor mutations are introduced in the new generation. This process is parameterized by the number of mutable individuals and genes within each one. As there not exists a limit on the

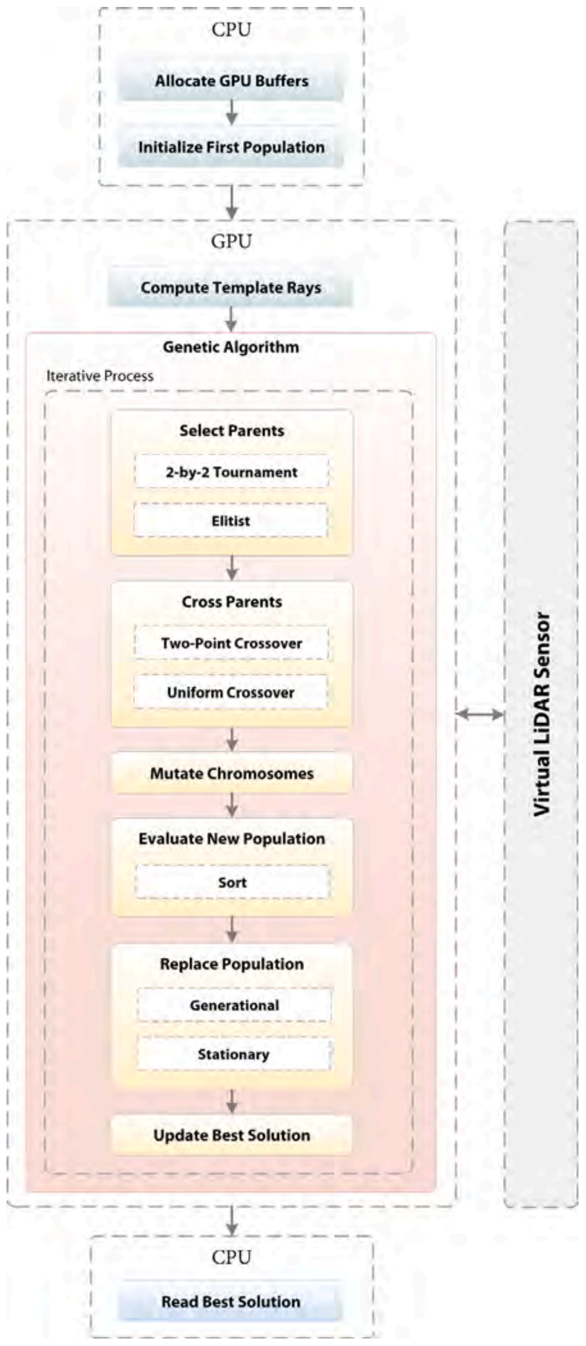

Fig. 9. Overview of the genetic algorithm steps to evaluate the best combination of spatial LiDAR setups.

number of activated bits, mutations are either performed by activating or deactivating random bits.

Evaluation of new population. We compute the fitness of new solutions for later stages based on the number of reached polygons. Consequently, solutions that cover more polygons are more likely to reproduce in the following iterations. However, this metric leads the population to activate the whole set of locations. For this reason, the previously proposed F_1 metric is modified according to Eq. (9) by also accounting for the number of activated bits within a chromosome (c_g). As a result, the generic algorithm is oriented towards reaching the maximum number of polygons with the minimum amount of LiDAR scans.

$$F_1 = \frac{\sum_{s=1}^{|S|} 1[t_s \in \{L_1, L_2, \dots, L_k\}]}{\sum_{g=1}^{n} c_g} \tag{9}$$

Replace population. New and previous populations are here mixed following a generational or stationary method. The generational approach replaces parents with their offspring, whereas the stationary algorithm replaces the worst individuals with the new population whether they improve current solutions.

Update best solution. It is noteworthy that even this minor operation must be performed on the GPU, as the reading transfer would lead to a huge delay in the GA response time.

3.7. *Interactive tools*

This section presents the interface tools provided to the user to facilitate the adjustment of the scene and the algorithm set-up.

3.7.1. *Level selection*

The described P4S method is not restricted to single-floor environments. Instead, we have evaluated it with multiple floors. However, the planning ought to be performed individually for each one. To easier the selection of floor-labelled polygons, an interactive tool is provided within the application. For that purpose, we added a ray-casting tool to generate rays whose direction is given by the user's viewpoint and a 3D position. This position is computed by unprojecting the camera matrix to a 2D point within the application canvas. The first collided model is pushed into the ground list. The ray-casting process is accelerated using the BVH data structure, thus solving it with unnoticeable latency. With this regard, Eq. (10) defines how to unproject a 2D point from the canvas to 3D.

$$p_{3D} = (P \cdot V \cdot M)^{-1} \cdot \left[\frac{p_{2D_x}}{c_x}2 - 1, \frac{p_{2D_y}}{c_y}2 - 1, 0, 1\right]^T \tag{10}$$

$$p_{3D} = \frac{p_{3D}}{p_{3D_w}} \tag{11}$$

where (p_{2D_x}, p_{2D_y}) is the canvas point, P, V are the camera projection and view matrices, respectively, (c_x, c_y) is the canvas size, and p_{3D} is a 4-tuple describing the projection coordinates.

4. Results and discussion

We have evaluated the proposed framework using three different scenes obtained from BIM management software (Fig. 10: Basement, School and Office building). These models were exported as OBJ files ranging from 130k polygons to 2.7M. Therefore, traversing the BVH takes a substantial portion of time during the optimization. Although there exist a few studies concerning P4S in 3D, their solution is not publicly published. Therefore, we compared our GPU-based solution with the multi-core CPU-based approach. The latter version is implemented using OpenMP (Open Multi-Processing), a multi-threading framework for CPU-based processes. Despite this, the LiDAR evaluation is very time-consuming on the CPU and therefore, both versions evaluate LiDAR scanning in the GPU.

The exploring parameters for the three local searches are set as defined in Table 1. On the other hand, LiDAR parameters are detailed in Table 2. All measurements were performed on a PC with AMD Ryzen Threadripper 3970X 3.6 GHz, 256 GB RAM, two Nvidia RTX A6000 GPU and Windows 10 OS. The massively parallel methods in the GPU were implemented in GLSL using OpenGL (Open Graphics Library).

This section is organized as follows. First, the performance of different LS algorithms is shown based on the F_1 and F_2 metrics. Then, the complete pipeline is evaluated by combining LS and GA. We also aim to compare GPU and CPU approaches to massively launch GA algorithms. Finally, the impact of height variability on LiDAR optimizations, i.e., 3D planning for scanning, is also explained.

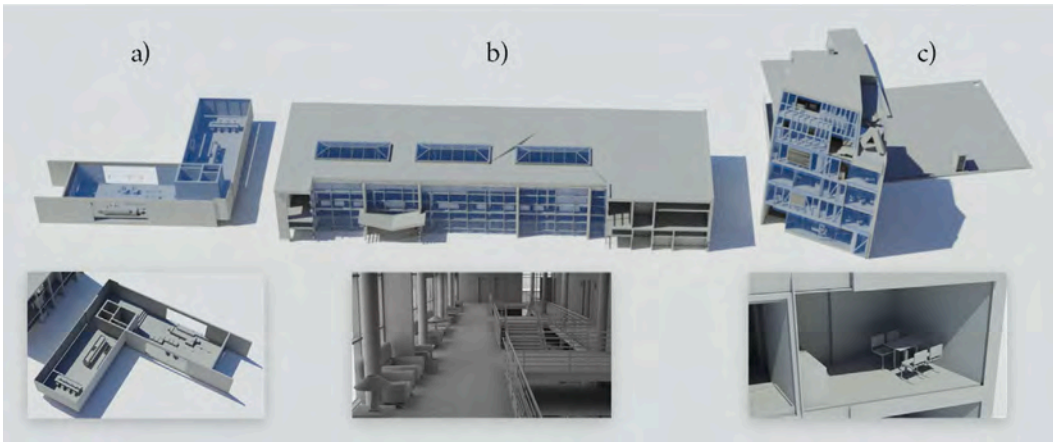

Fig. 10. Three different environments from Autodesk Revit® applied to our evaluation. (a) Basement with 130k triangles (20 × 4 × 21 m), (b) school with 500K triangles (44 × 24 × 32 m) and (c), office building with 2.7M triangles (16 × 8 × 49 m).

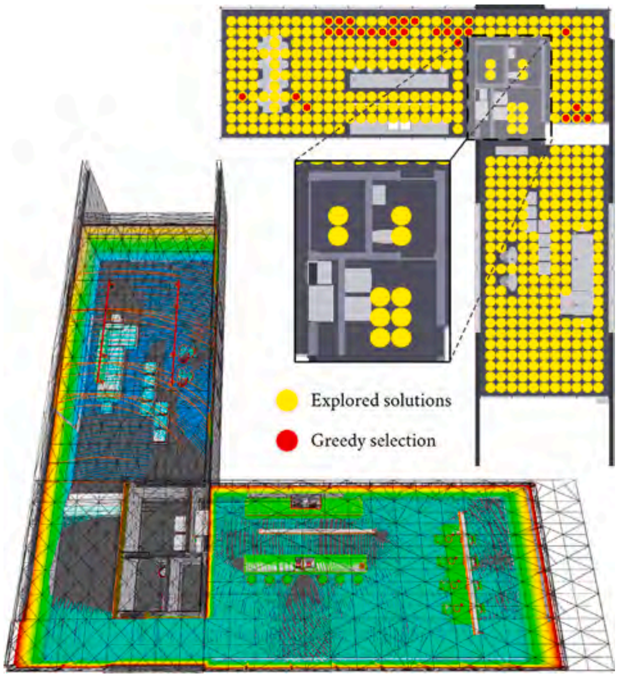

Fig. 11. Selection of the best 30 LiDAR locations using the greedy approach in the Basement scene.

Table 1
Overall attributes concerning the three cited local search algorithms.

Attributes	Value
Starting solutions	Uniform grid sampling
Final LS solutions	No limit
Voxel size	0.5 m × 0.4 m
Max. iterations	60
Max. iterations without improv.	10
Neighbourhood	Discrete (16)
Δ Neighbourhood	0.05 m
T_0	450 °C
T_z	0.8 T_{z-1}

Table 2
Specifications of LiDAR sensor during optimization, following the commercial device HDL-64E. Attributes with * have been adapted either to reduce the optimization latency or to account for accuracy metrics (e.g., range).

Attributes	Value
Resolution	4500 × 64 beams
Max. bounces	1*
Max. range	5 m*
Coverage	360° × 26.9° (−24.9°–2°)
Height coverage	[0.5, 2] m
Min. xz distance to items	0.4 m
Vertical checks	r ← 0.2 m, n ← 16 rays

4.1. Performance of local searches

This section is focused on showing the improvement of the F_2 metric in three different LS methods: greedy, traditional LS, simulated annealing and tabu search. The evaluation of the F_2 metric for each initial solution is depicted regardless of the best solution observed until such iteration. We used the Basement scene during this evaluation. Despite the improvement observed after using LS, the relevance of GA selection after this first phase is depicted in Fig. 11. It shows the narrowing phase of a Greedy approach that selects the best $k \leftarrow 30$ locations (red) in an environment of variable geometrical complexity among different areas, guided by the minimization of the F_2 metric. Hence, most of the selected locations are surrounded by dense geometry. Still, it manages to cover most of the scene. Similarly, small enclosed rooms, as the zoom-in of Fig. 11, are not scanned due to their smaller number of polygons.

From this baseline, Fig. 12 depicts the improvements of initial solutions with different LS algorithms. For that purpose, we show the F_1 and F_2 fitness values achieved by individual solutions during a fixed number of iterations. However, some explorations may finish earlier if no better solutions are found. Since we do not include GA selection yet, optimized locations are filtered according to a regular grid that limits the number of solutions per cell to one.

Local search. As expected from the description of LS, the exploration of initial solutions using LS finishes early, especially for locations that start with low F_2 values. The algorithm does not allow moving to worse solutions, thus favouring that the maximum number of iterations is achieved early. Also note that moves to neighbours are controlled by a small factor, otherwise the initial grid sampling turns into a random sampling. Due to the behaviour of LS, most locations would converge into positions close to walls as regarded by Soudarissanane and Lindenbergh [11], since they maximize F_1 and collide walls with low incidence angle.

Simulated annealing. As opposed to LS, SA is able to temporarily worsen the location fitness and allows exploring a wider area in xz and y. Hence, most of the initial locations reach lower values of F_2 than

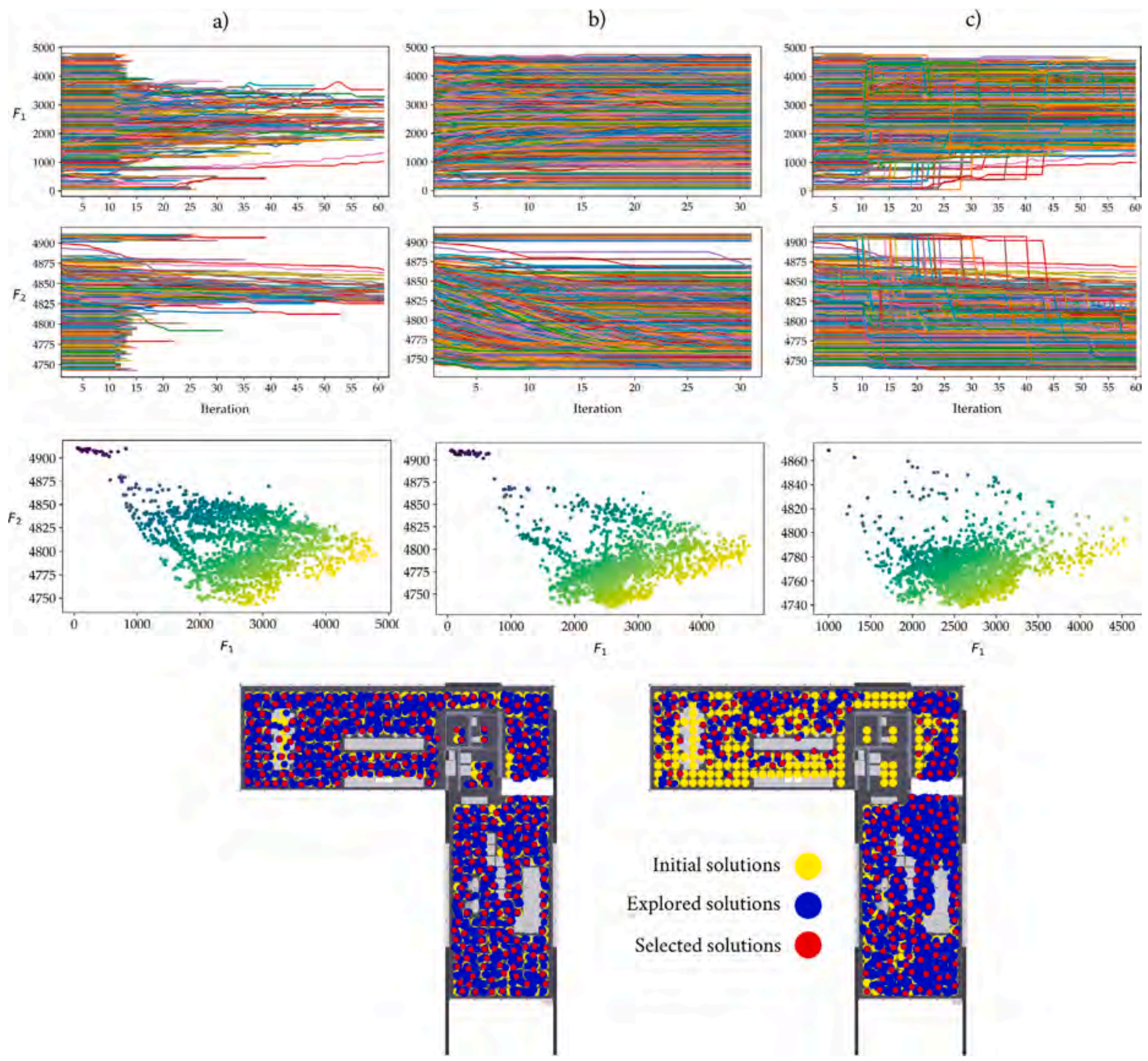

Fig. 12. Optimization performed by (a) naive LS, (b) SA and (c) TS, showing both F_1 and F_2 results through their iterative improvement and a scatter plot. The bottom images show the final locations of solutions optimized by SA and TS. Then, these are organized according to a regular grid and filtered by selecting only one solution per subdivision.

LS, as shown in the scatter plot. Implicitly, initial solutions move to locations that cover a larger number of polygons due to F_1 being also integrated into F_2's formula. Unlike LS, early-finished explorations are not as frequent. Despite LOC being included in F_2, the line graph shows iterations that reach a lower number of collided polygons than previous solutions. It is not necessarily correlated to SA allowing worse solutions, since coverage is not a primary metric in this first stage. Instead, it is a consequence of moves that improve accuracy at expense of slightly worsening coverage. For that reason, the subsequent GA stage is aimed at providing better coverage.

Tabu search. By avoiding previous moves and re-initializing when getting stuck, initial solutions are slightly more optimal regarding F_2 (not for F_1) than those obtained by previous algorithms. However, they are not uniformly sparsed as they (1) tend to show a preference for areas with higher polygon density and (2) random re-initialization favours this. Despite it being desirable, the local improvement of SA is preferred over TS as it offers a wider range of possibilities to the later GA, whereas TS offers immediate results whether we need to select the best k locations. Also, note that the steeper profiles are due to the re-initialization phase implemented by selecting new random points. Without re-initialization, TS turns into a more strict LS. Hence, the main aim of TS was to show the impact of re-initializing.

4.2. Path performance

The main challenge of optimizing the LiDAR set-up is to cover the whole building level, including enclosed rooms. However, the described LS approaches optimize local metrics. Thus, the sorting and narrowing of locations by their F_2 result only guarantee to select points that densely acquire their surrounding area. Therefore, the final selection is performed by genetic algorithms guided by a global metric measuring the polygon coverage. The evaluated genetic algorithm is launched using the values as shown in Table 3. We also aim to evaluate the results of connected nodes, i.e., LS + GA, Simulated Annealing + GA, etc. Besides coverage, the implicit reduction of LiDAR locations during the optimization is also observed during these tests.

The results from three different combinations of LS and genetic algorithms are depicted in Fig. 13, using a single level of the School environment. This scenario includes multiple enclosed rooms that cannot be scanned from central areas, as well as gaps in the central area that allows scanning lower building levels. In this regard, the tabu search (c) obtained a lower number of points, though most of them are gathered in the main room, as happened with locally improved solutions being translated to areas with higher geometrical complexity. In contrast to the tabu search, LS and SA provide a higher variety of locations, thereby favouring the selection of a higher number of locations and providing

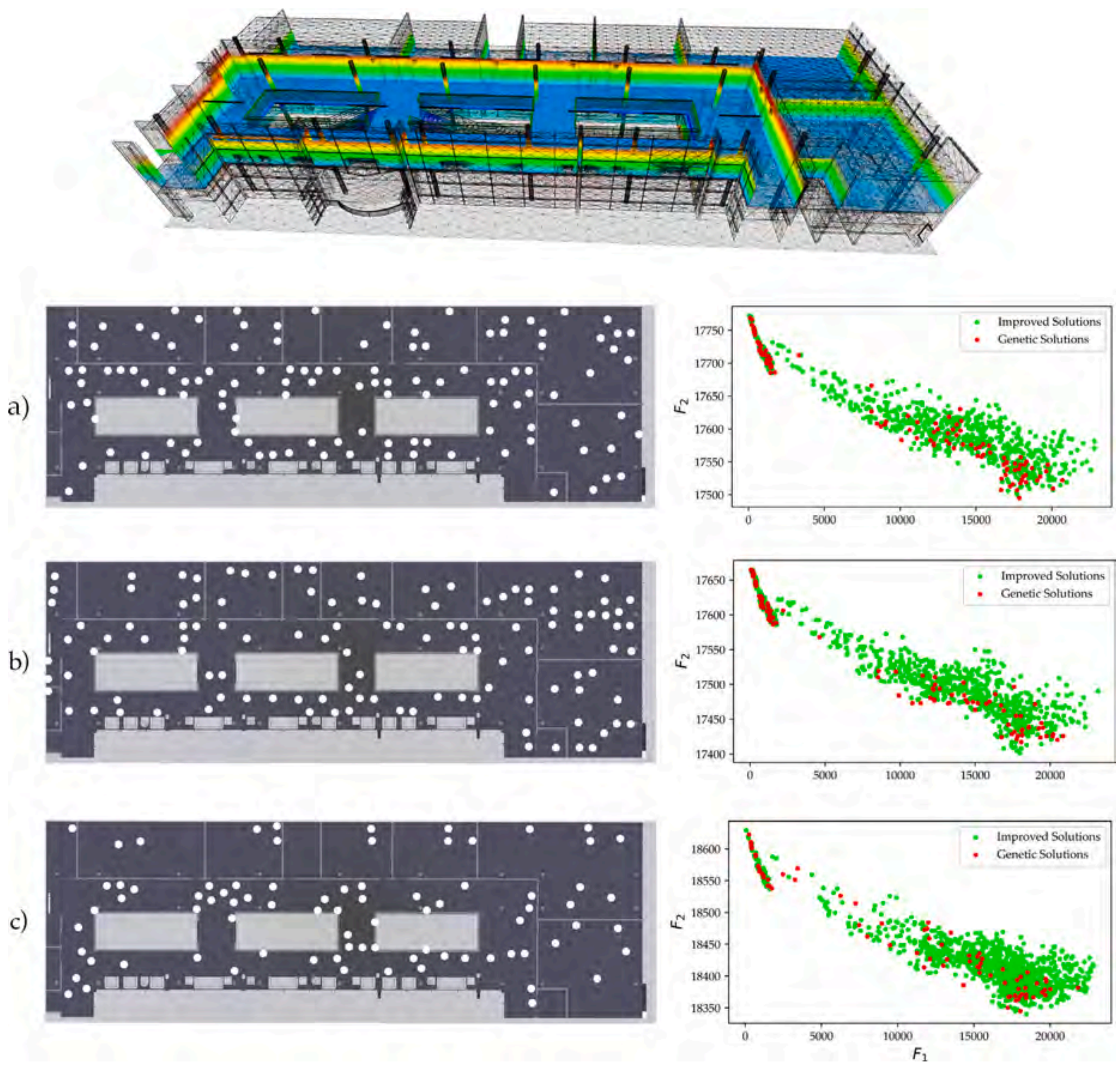

Fig. 13. Results of combining local searches and genetic algorithms. The left images depict the final LiDAR distribution, whereas the right images show the improvement of intermediate results with regard to the proposed metric ($Y \leftarrow$ Minimization, $X \leftarrow$ Maximization). a), b) and c) refer to LS + GA, SA + GA and TS + GA combinations.

better coverage of enclosed rooms. Also, note that the LiDAR range was limited to $r \leftarrow 5$ m to account for accuracy loss due to the distance. Therefore, higher values of r derive into a less dense selection. Despite LS and SA providing similar results, the GA selection using SA managed to scan the small rooms on the left and right sides due to its wider exploration. Nevertheless, the GA performance in terms of F_1 and F_2 is similar for the three algorithms, as regarded in the scatter plots of Fig. 13. It can be concluded that, despite the importance of selecting scans with GA, it is significantly conditioned by the previous local search step.

Table 3
Parameters of the genetic algorithm launched in this stage.

Attributes	Value
Max. evaluations	50K
Population size	500
Number of parents	250
P(Chromosome mutation)	0.02
P(Gen mutation)	0.1
Stagnant population (re-init.)	80%
Stuck at local minima (re-init.)	30 it. without improving
Crossover operator	Two points
Selection operator	2-tuple tournament
Replacement	Stationary

4.3. Level of overlap

Previous LiDAR locations are selected according to the F_1 and F_2 metrics to fit LOC, LOD and LOA requirements, since solving GA and LS algorithms with a single objective function is straightforward. Thus, LOO is considered once the optimal set of solutions is selected, as subsequently added locations are aimed at improving rather than providing an optimal solution. In this subsection, we evaluate how the proposed overlap method affects the final result. To facilitate the visualization of LOO, a greedy approach is used to (1) sample the scenario and (2) narrow it to a few solutions scattered by controlling the density of solutions. Accordingly, Fig. 14 shows the initial level of overlapping as well as the improvement after requiring an overlap of 20%, 40% and 60%. Below, the LiDAR distribution is obtained by limiting the sensor range to 1.5 m and 1 m, respectively. Also, the last test was further stressed by scattering the solutions with a minimum distance of (6 - ϵ) m. Consequently, the minimum number of solutions to reach each other is 2, though it varies according to their distance.

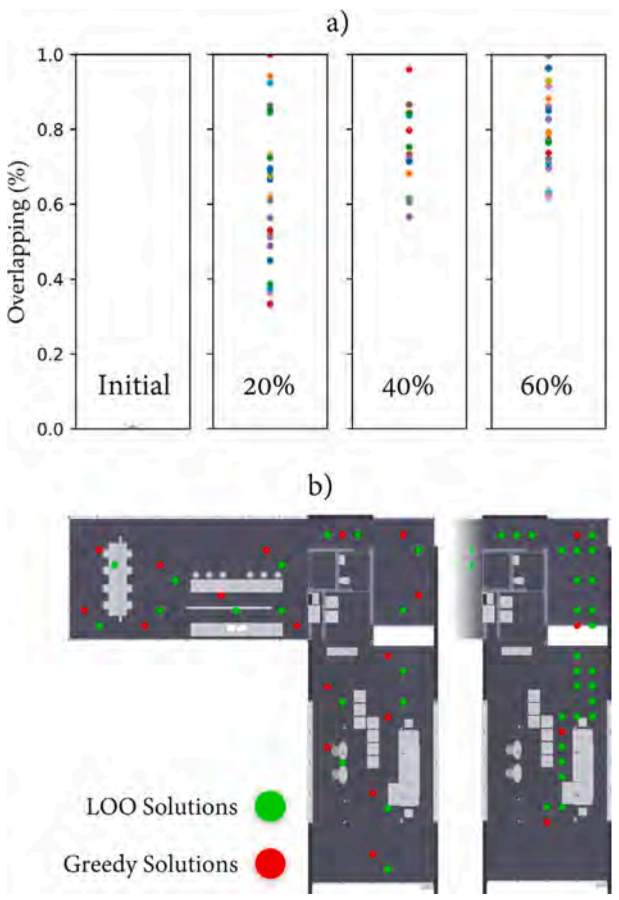

Fig. 14. (a) Initial overlap and the results obtained after requiring an overlap of 20%, 40% and 60%. (b) Rendering of LiDAR scans selected by a Greedy approach as well as those selected by the proposed method to enhance the overlap.

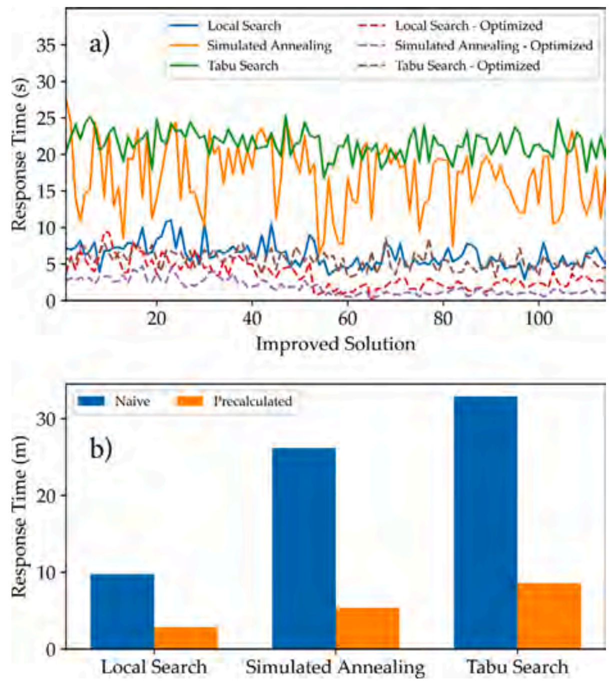

Fig. 15. (a) Average response time per solution for every LS algorithm with two versions: initial and optimized, and (b) summation of response time from previous configurations.

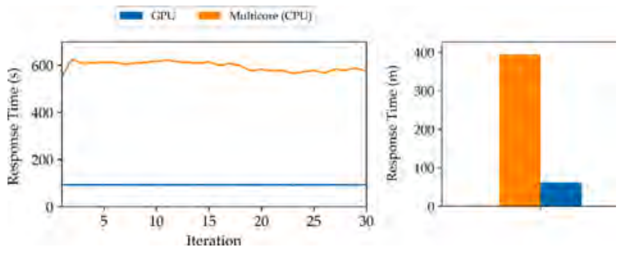

Fig. 16. Response time of genetic algorithms implemented as CPU (multi-core) and GPU solutions. First, the response time is shown per GA population, and finally, the response time is summed for each sort of optimization.

Initial solutions were not improved by means of local searches, thus displaying a grid-like pattern.

4.4. Response time

The main drawback of local searches is the sequential evaluation of LiDAR scans. Our proposal is conditioned by GPU LiDAR evaluations, as they cannot be solved in parallel. The performance of the GA algorithm was improved by solving the whole simulation of a chromosome at once since it is composed of several locations. However, changes in locations must be evaluated individually during a local search. On the other hand, improvements based on pre-calculations are more feasible at expense of higher CPU-memory usage. Therefore, some optimizations and implementation details are here discussed to provide an efficient solution.

The current LS approach checks the validity and computes the metric values of every new solution, regardless of the possibility that it may have been previously checked. It seldom occurs with random initialization, though it is known to occur frequently with uniformly sampled locations due to the constant step length. Therefore, it is possible to reduce the latency by pre-calculating both the validity and metric values of grid voxels whose length depends on the neighbour step length. Still, the latency remains the same for a greedy approach. Fig. 15 shows the average and global response time for the three LS approaches. The Building scene was used to conduct this evaluation, using the configuration in Table 1. The first chart shows the average response time per LiDAR location. Hence, latency varies depending on the number of iterations. The global latency sums the search as well as the pre-calculation latency. Accordingly, we can conclude that this approach significantly speed-ups the search, except in those algorithms that get easily stuck in local optima or cannot reinitialize.

During the selection stage, different chromosomes are evaluated synchronously. Parallelism is present in other stages: initialization, parent selection, crossover, etc. Hence, we evaluated whether it was effective to implement the genetic algorithm pipeline on the GPU. Initially, it is implemented as a multi-thread solution on the CPU. From Fig. 15 we can observe that the GPU-based solution is significantly faster. It mainly occurs due to data being processed in the GPU for the whole GA pipeline, instead of transferring it before and after evaluating the solution's fitness (see Fig. 16).

4.5. Variable height LiDAR

Besides occlusion-aware 3D P4S, another contribution of our work is the estimation of the most appropriate height when scanning. Fig. 17 shows the results of F_1 and F_2 metrics by using (1) fixed height or (2) variable height within the range [1, 2] m. In both cases, the scenario is uniformly subsampled every 0.1 m. Accordingly, the first search is performed over a 2D grid, regardless of being applied over a 3D scene, whereas the second one is 3D. The results are obtained using a test

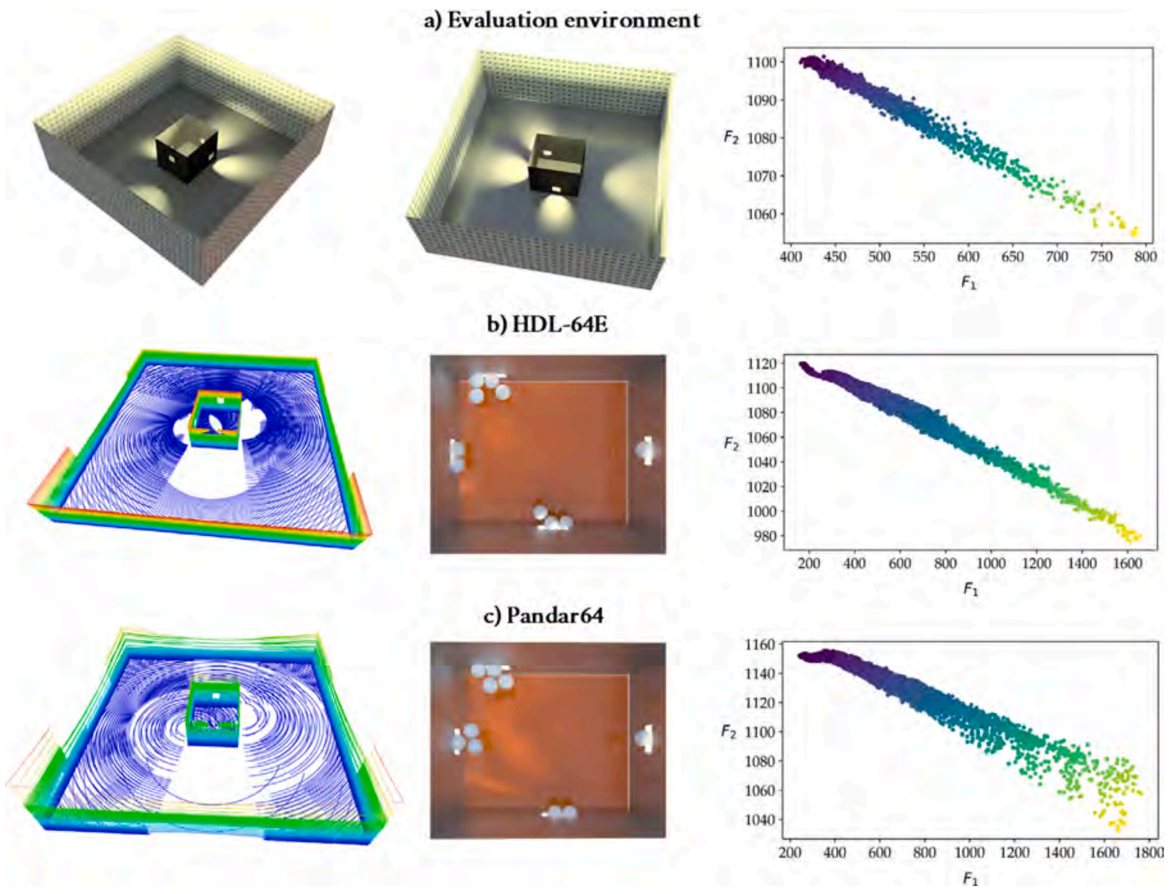

Fig. 17. (a) An environment designed for testing the benefits of variable height, and (b), (c) locations selected by a Greedy algorithm using an HDL-64E and Pandar64 LiDAR sensors. The scatter plots show the values of F_1 and F_2 for each initial solution. The first one shows the results of a LiDAR with a fixed height.

environment composed of four outer walls and four inner walls with one hole per wall, each one at different height, as depicted in Fig. 17. The LiDAR was first configured as a Velodyne HDL-64E, and then, as a Pandar64 with a wider vertical Field of View (FOV) and non-uniform resolution. Instead of applying an LS, a Greedy approach is used to evaluate a fine subdivision of the space that is subsequently narrowed to ten placements. As observed in Fig. 17, all the locations selected by the Greedy approach are placed next to wall holes, thereby allowing to reach more polygons. Despite being guided by the F_2 metric, these locations are mainly selected according to the number of reached polygons, thus showing the inclusion of F_1 in F_2.

5. Conclusions and future work

In this work, a P4S methodology was described to select the best n scanning locations in complex 3D environments, taking advantage of GPU-based spatial queries and LiDAR scans. Instead of applying a single metaheuristic algorithm, as in most of the revised work in the literature, our pipeline is split into two different phases: local optimizations and selection. Regarding objective functions, the described metrics do not rely on manually set values. Instead, they are shaped with smoothed boundaries and some of them, such as LOA, are improved by considering several criteria that enhance multiple metrics at once. Also, the optimal value of n is implicitly explored during the optimization guided by the modified LOC metric. Finally, algorithms were optimized as multi-core CPU and GPU in the case of genetic algorithms. The results show that our method is capable of constructing a LiDAR set-up distribution to scan environments with significant occlusion by discovering the optimal sensor height.

Despite being accelerated in CPU and GPU, the proposed methodology is still time-consuming due to the dimensionality of both 3D environments and LiDAR scans guided by specifications of commercial sensors. Hence, pre-calculating the fitness of uniformly scattered locations ought to be further explored, at expense of storage volume. Therefore, compression may be a key factor due to the similarity of close scans. Finally, continuous paths should also be considered in addition to discrete locations to cover other kinds of LiDAR sensors, such as mobile or aerial scanning.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

The authors do not have permission to share data.

Acknowledgements

This result has been partially supported through the research projects TIN2017-84968-R and RTI2018-099638-B-I00 funded by MCIN/AEI/10.13039/501100011033/ and ERDF funds “A way of doing Europe", as well as by the Spanish Ministry of Science, Innovation and Universities via a doctoral grant to the first author (FPU19/00100).

References

[1] J. Pandžić, M. Pejić, B. Božić, V. Erić, Error model of direct georeferencing procedure of terrestrial laser scanning, Autom. Constr. 78 (2017) 13–23, http://dx.doi.org/10.1016/j.autcon.2017.01.003.

[2] M.H. Shariq, B.R. Hughes, Revolutionising building inspection techniques to meet large-scale energy demands: A review of the state-of-the-art, Renew. Sustain. Energy Rev. 130 (2020) 109979, http://dx.doi.org/10.1016/j.rser.2020.109979.

[3] E. Guisado-Pintado, D.W.T. Jackson, D. Rogers, 3D mapping efficacy of a drone and terrestrial laser scanner over a temperate beach-dune zone, Geomorphology 328 (2019) 157–172, http://dx.doi.org/10.1016/j.geomorph.2018.12.013.

[4] H. Mitasova, E. Hardin, M.F. Overton, M.O. Kurum, Geospatial analysis of vulnerable beach-foredune systems from decadal time series of lidar data, J. Coast. Conserv. 14 (3) (2010) 161–172, http://dx.doi.org/10.1007/s11852-010-0088-1.

[5] S. Kuutti, R. Bowden, Y. Jin, P. Barber, S. Fallah, A survey of deep learning applications to autonomous vehicle control, IEEE Trans. Intell. Transp. Syst. 22 (2) (2021) 712–733, http://dx.doi.org/10.1109/TITS.2019.2962338.

[6] F. Banfi, The integration of a scan-to-HBIM process in BIM application: the development of an add-in to guide users in autodesk revit, ISPRS - Int. Arch. Photogramm. Remote Sens. Spatial Inf. Sci. XLII-2/W11 (2019) 141–148, http://dx.doi.org/10.5194/isprs-archives-XLII-2-W11-141-2019.

[7] N. Ham, B.I. Bae, O.K. Yuh, Phased reverse engineering framework for sustainable cultural heritage archives using laser scanning and BIM: the case of the hwanggungwoo (seoul, korea), Sustainability 12 (19) (2020) 8108, http://dx.doi.org/10.3390/su12198108.

[8] M. Andriasyan, J. Moyano, J.E. Nieto-Julián, D. Antón, From point cloud data to building information modelling: an automatic parametric workflow for heritage, Remote Sens. 12 (7) (2020) 1094, http://dx.doi.org/10.3390/rs12071094.

[9] F. Poux, R. Billen, A Smart Point Cloud Infrastructure for intelligent environments, in: Laser Scanning: An Emerging Technology in Structural Engineering, CRC Press, 2019, pp. 127–149.

[10] A. Warchoł, the concept of LIDAR data quality assessment in the context of BIM modeling, ISPRS - Int. Arch. Photogramm. Remote Sens. Spatial Inf. Sci. XLII-1/W2 (2019) 61–66, http://dx.doi.org/10.5194/isprs-archives-XLII-1-W2-61-2019.

[11] S. Soudarissanane, R. Lindenbergh, Optimizing terrestrial laser scanning measurement set-up, ISPRS - Int. Arch. Photogramm. Remote Sens. Spatial Inf. Sci. XXXVIII-5/W12 (2012) 127–132, http://dx.doi.org/10.5194/isprsarchives-XXXVIII-5-W12-127-2011.

[12] H. Macher, T. Landes, P. Grussenmeyer, From point clouds to building information models: 3D semi-automatic reconstruction of indoors of existing buildings, Appl. Sci. 7 (10) (2017) 1030, http://dx.doi.org/10.3390/app7101030.

[13] V. Pătrăucean, I. Armeni, M. Nahangi, J. Yeung, I. Brilakis, C. Haas, State of research in automatic as-built modelling, Adv. Eng. Inform. 29 (2) (2015) 162–171, http://dx.doi.org/10.1016/j.aei.2015.01.001.

[14] Rocha, Mateus, Fernández, Ferreira, A scan-to-BIM methodology applied to heritage buildings, Heritage 3 (1) (2020) 47–67, http://dx.doi.org/10.3390/heritage3010004.

[15] J. Moyano, C.P. Odriozola, J.E. Nieto-Julián, J.M. Vargas, J.A. Barrera, J. León, Bringing BIM to archaeological heritage: Interdisciplinary method/strategy and accuracy applied to a megalithic monument of the Copper Age, J. Cult. Herit. 45 (2020) 303–314, http://dx.doi.org/10.1016/j.culher.2020.03.010.

[16] C. Gollob, T. Ritter, A. Nothdurft, Comparison of 3D point clouds obtained by terrestrial laser scanning and personal laser scanning on forest inventory sample plots, Data 5 (4) (2020) 103, http://dx.doi.org/10.3390/data5040103.

[17] P. Rodríguez-Gonzálvez, B. Jiménez Fernández-Palacios, A.L. Muñoz-Nieto, P. Arias-Sanchez, D. Gonzalez-Aguilera, Mobile LiDAR system: new possibilities for the documentation and dissemination of large cultural heritage sites, Remote Sens. 9 (3) (2017) 189, http://dx.doi.org/10.3390/rs9030189.

[18] A. Bienert, L. Georgi, M. Kunz, H.G. Maas, G. Von Oheimb, Comparison and combination of mobile and terrestrial laser scanning for natural forest inventories, Forests 9 (7) (2018) 395, http://dx.doi.org/10.3390/f9070395.

[19] T.H. Kim, T.H. Park, Placement optimization of multiple lidar sensors for autonomous vehicles, IEEE Trans. Intell. Transp. Syst. 21 (5) (2020) 2139–2145, http://dx.doi.org/10.1109/TITS.2019.2915087.

[20] A. López, C.J.O. Anguita, F.R.F. Higueruela, A GPU-Accelerated LiDAR Sensor for Generating Labelled Datasets, The Eurographics Association, 2021, http://dx.doi.org/10.2312/ceig.20211360.

[21] L. Li, Z. Wei, J.K. Hao, K. He, Probability learning based tabu search for the budgeted maximum coverage problem, Expert Syst. Appl. 183 (2021) 115310, http://dx.doi.org/10.1016/j.eswa.2021.115310.

[22] H.E. Mohamadi, N. Kara, M. Lagha, Efficient algorithms for decision making and coverage deployment of connected multi-low-altitude platforms, Expert Syst. Appl. 184 (2021) 115529, http://dx.doi.org/10.1016/j.eswa.2021.115529.

[23] V. Roostapour, A. Neumann, F. Neumann, T. Friedrich, Pareto optimization for subset selection with dynamic cost constraints, Artificial Intelligence 302 (2022) 103597, http://dx.doi.org/10.1016/j.artint.2021.103597.

[24] C. Potthast, G.S. Sukhatme, A probabilistic framework for next best view estimation in a cluttered environment, J. Vis. Commun. Image Represent. 25 (1) (2014) 148–164, http://dx.doi.org/10.1016/j.jvcir.2013.07.006.

[25] M. Giorgini, S. Marini, R. Monica, J. Aleotti, Sensor-based optimization of terrestrial laser scanning measurement setup on GPU, IEEE Geosci. Remote Sens. Lett. 16 (9) (2019) 1452–1456, http://dx.doi.org/10.1109/LGRS.2019.2899681.

[26] A. Aryan, F. Bosché, P. Tang, Planning for terrestrial laser scanning in construction: A review, Autom. Constr. 125 (2021) 103551, http://dx.doi.org/10.1016/j.autcon.2021.103551.

[27] E. Wakisaka, S. Kanai, H. Date, Optimal laser scan planning for as-built modeling of plant renovations using mathematical programming, ISARC Proc. (2019) 91–98.

[28] D. Li, J. Liu, Y. Zeng, G. Cheng, B. Dong, Y.F. Chen, 3D model-based scan planning for space frame structures considering site conditions, Autom. Constr. 140 (2022) 104363, http://dx.doi.org/10.1016/j.autcon.2022.104363.

[29] M.J. Starek, T. Chu, H. Mitasova, R.S. Harmon, Viewshed simulation and optimization for digital terrain modelling with terrestrial laser scanning, Int. J. Remote Sens. 41 (16) (2020) 6409–6426, http://dx.doi.org/10.1080/01431161.2020.1752952.

[30] C. Zhang, V.S. Kalasapudi, P. Tang, Rapid data quality oriented laser scan planning for dynamic construction environments, Adv. Eng. Inform. 30 (2) (2016) 218–232, http://dx.doi.org/10.1016/j.aei.2016.03.004.

[31] M. Heidari Mozaffar, M. Varshosaz, Optimal placement of a terrestrial laser scanner with an emphasis on reducing occlusions, Photogramm. Rec. 31 (156) (2016) 374–393, http://dx.doi.org/10.1111/phor.12162.

[32] E. Latimer, D. Latimer, R. Saxena, C. Lyons, L. Michaux-Smith, S. Thayer, Sensor space planning with applications to construction environments, in: IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA '04. 2004, Vol. 5, 2004, pp. 4454–4460, http://dx.doi.org/10.1109/ROBOT.2004.1302419.

[33] H. Chen, W. Guan, S. Li, Y. Wu, Indoor high precision three-dimensional positioning system based on visible light communication using modified genetic algorithm, Opt. Commun. 413 (2018) 103–120, http://dx.doi.org/10.1016/j.optcom.2017.12.045.

[34] F. Jia, D. Lichti, A comparison of simulated annealing, genetic algorithm and particle swarm optimization in optimal first-order design of indoor TLS networks, in: ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences, Vol. IV-2-W4, Copernicus GmbH, 2017, pp. 75–82, http://dx.doi.org/10.5194/isprs-annals-IV-2-W4-75-2017.

[35] J. Ahn, K. Wohn, Interactive scan planning for heritage recording, Multimedia Tools Appl. 75 (7) (2016) 3655–3675, http://dx.doi.org/10.1007/s11042-015-2473-0.

[36] Y. Pehlivanoglu, P. Pehlivanoglu, An enhanced genetic algorithm for path planning of autonomous UAV in target coverage problems, Appl. Soft Comput. 112 (2021) http://dx.doi.org/10.1016/j.asoc.2021.107796.

[37] V. Roberge, M. Tarbouchi, Parallel algorithm on GPU for wireless sensor data acquisition using a team of unmanned aerial vehicles, Sensors 21 (20) (2021) 6851, http://dx.doi.org/10.3390/s21206851.

[38] R. Islambouli, S. Sharafeddine, Optimized 3D deployment of UAV-mounted cloudlets to support latency-sensitive services in IoT networks, IEEE Access 7 (2019) 172860–172870, http://dx.doi.org/10.1109/ACCESS.2019.2956150.

[39] M. Pereira, D. Silva, V. Santos, P. Dias, Self calibration of multiple LIDARs and cameras on autonomous vehicles, Robot. Auton. Syst. 83 (2016) 326–337, http://dx.doi.org/10.1016/j.robot.2016.05.010.

[40] K. Na, J. Byun, M. Roh, B. Seo, Fusion of multiple 2D LiDAR and RADAR for object detection and tracking in all directions, in: 2014 International Conference on Connected Vehicles and Expo, ICCVE, 2014, pp. 1058–1059, http://dx.doi.org/10.1109/ICCVE.2014.7297512.

[41] L.d.P. Veronese, A. Ismail, V. Narayan, M. Schulze, An accurate and computational efficient system for detecting and classifying ego and sides lanes using LiDAR, in: 2018 IEEE Intelligent Vehicles Symposium, IV, 2018, pp. 1476–1483, http://dx.doi.org/10.1109/IVS.2018.8500434.

[42] Z.J. Wang, Z.H. Zhan, J. Zhang, Solving the energy efficient coverage problem in wireless sensor networks: a distributed genetic algorithm approach with hierarchical fitness evaluation, Energies 11 (12) (2018) 3526, http://dx.doi.org/10.3390/en11123526.

[43] T. Voegtle, I. Schwab, T. Landes, Influences of different materials on the measurement of a Terrestrial Laser Scanner (TLS), in: Proc. of the XXI Congress, the International Society for Photogrammetry and Remote Sensing, ISPRS2008, Vol. 37, 2008.

[44] G. Lee, J. Cheon, I. Lee, Validation of LIDAR calibration using a LIDAR simulator, ISPRS - Int. Arch. Photogramm. Remote Sens. Spatial Inf. Sci. XLIII-B1-2020 (2020) 39–44, http://dx.doi.org/10.5194/isprs-archives-xliii-b1-2020-39-2020.

[45] V. Méndez, H. Catalán, J.R. Rosell-Polo, J. Arnó, R. Sanz, LiDAR simulation in modelled orchards to optimise the use of terrestrial laser scanners and derived vegetative measures, Biosyst. Eng. 115 (1) (2013) 7–19, http://dx.doi.org/10.1016/j.biosystemseng.2013.02.003.

[46] J. Iqbal, R. Xu, S. Sun, C. Li, Simulation of an autonomous mobile robot for LiDAR-based in-field phenotyping and navigation, Robotics 9 (2) (2020) 46, http://dx.doi.org/10.3390/robotics9020046.

[47] F. Westling, M. Bryson, J. Underwood, SimTreeLS: Simulating aerial and terrestrial laser scans of trees, 2020, arXiv:2011.11954 [cs, eess].

[48] S.D. Brown, D.D. Blevins, J.R. Schott, Time-gated topographic LIDAR scene simulation, in: G.W. Kamerman (Ed.), Laser Radar Technology and Applications X, SPIE, 2005, pp. 342–353, http://dx.doi.org/10.1117/12.604326.

[49] B. Lohani, R. Mishra, Generating LiDAR data in laboratory: LiDAR simulator, Int. Arch. Photogramm. Remote Sens. 52 (2007).

[50] A. Hovi, I. Korpela, Real and simulated waveform-recording LiDAR data in juvenile boreal forest vegetation, Remote Sens. Environ. 140 (2014) 665–678, http://dx.doi.org/10.1016/j.rse.2013.10.003.

[51] J.-P. Gastellu-Etchegorry, T. Yin, N. Lauret, E. Grau, J. Rubio, B.D. Cook, D.C. Morton, G. Sun, Simulation of satellite, airborne and terrestrial LiDAR with DART (I): Waveform simulation with quasi-Monte Carlo ray tracing, Remote Sens. Environ. 184 (2016) 418–435, http://dx.doi.org/10.1016/j.rse.2016.07.010.

[52] T. Yin, N. Lauret, J.P. Gastellu-Etchegorry, Simulation of satellite, airborne and terrestrial LiDAR with DART (II): ALS and TLS multi-pulse acquisitions, photon counting, and solar noise, Remote Sens. Environ. 184 (2016) 454–468, http://dx.doi.org/10.1016/j.rse.2016.07.009.

[53] T. Yun, L. Cao, F. An, B. Chen, L. Xue, W. Li, S. Pincebourde, M.J. Smith, M.P. Eichhorn, Simulation of multi-platform LiDAR for assessing total leaf area in tree crowns, Agricult. Forest Meteorol. 276–277 (2019) 107610, http://dx.doi.org/10.1016/j.agrformet.2019.06.009.

[54] P. Chen, C. Jamet, Z. Mao, D. Pan, OLE: a novel oceanic lidar emulator, IEEE Trans. Geosci. Remote Sens. (2020) 1–15, http://dx.doi.org/10.1109/tgrs.2020.3035381.

[55] T. Zohdi, Rapid simulation-based uncertainty quantification of flash-type time-of-flight and lidar-based body-scanning processes, Comput. Methods Appl. Mech. Engrg. 359 (2020) 112386, http://dx.doi.org/10.1016/j.cma.2019.03.056.

[56] N. Peinecke, T. Lueken, B.R. Korn, Lidar simulation using graphics hardware acceleration, in: 2008 IEEE/AIAA 27th Digital Avionics Systems Conference, IEEE, 2008, pp. 4.D.4–1–4.D.4–8, http://dx.doi.org/10.1109/dasc.2008.4702838.

[57] D. Meister, J. Bittner, Parallel locally-ordered clustering for bounding volume hierarchy construction, IEEE Trans. Vis. Comput. Graph. 24 (3) (2018) 1345–1353, http://dx.doi.org/10.1109/tvcg.2017.2669983.

[58] R. Marques, C. Bouville, K. Bouatouch, Optimal sample weights for hemispherical integral quadratures, Comput. Graph. Forum 38 (1) (2019) 59–72, http://dx.doi.org/10.1111/cgf.13392.

[59] L. Kocis, W.J. Whiten, Computational investigations of low-discrepancy sequences, ACM Trans. Math. Software 23 (2) (1997) 266–294, http://dx.doi.org/10.1145/264029.264064.

[60] J.B. Burkardt, The halton quasi monte carlo (QMC) sequence, 2010.

[61] J. Więckowski, B. Kizielewicz, J. Kołodziejczyk, Finding an approximate global optimum of characteristic objects preferences by using simulated annealing, in: I. Czarnowski, R.J. Howlett, L.C. Jain (Eds.), Intelligent Decision Technologies, in: Smart Innovation, Systems and Technologies, Springer, Singapore, 2020, pp. 365–375, http://dx.doi.org/10.1007/978-981-15-5925-9_31.

[62] M. Vannucci, V. Colla, S. Cateni, Genetic operators impact on genetic algorithms based variable selection, in: I. Czarnowski, R.J. Howlett, L.C. Jain (Eds.), Intelligent Decision Technologies, in: Smart Innovation, Systems and Technologies, Springer, Singapore, 2020, pp. 211–221, http://dx.doi.org/10.1007/978-981-15-5925-9_18.