This study advances indoor environment modeling by focusing on the optimal placement of sensors. Our approach involves creating a detailed environment model from a 3D point cloud by identifying spatial boundaries and furniture in indoor spaces, which are then represented as a series of polygons. To validate our method, we compare its performance against ground truth data, demonstrating high accuracy in both simple and complex environments. The core of our study is a comprehensive experiment that evaluates the effectiveness of three evolutionary nature-inspired genetic and three metaheuristic iterative optimization algorithms in solving the sensor placement problem in a complex environment scenario. We perform a statistical analysis to understand the impact of the choice of optimization algorithm and the number of sensors on the achieved spatial coverage. This analysis provides insights into the comparative effectiveness of various evolutionary algorithms in enhancing sensor network design within intricate indoor spaces. In particular, the Artificial Bee Colony algorithm consistently delivered superior results.
Advancing Indoor Environment Modeling: A Comparative Study of Evolutionary Algorithms for Optimal Sensor Placement

Marina Banov, Student Member, IEEE, Domagoj Pinčić, Kristijan Lenac, Member, IEEE, Diego Sušanj

Abstract—This study advances indoor environment modeling by focusing on the optimal placement of sensors. Our approach involves creating a detailed environment model from a 3D point cloud by identifying spatial boundaries and furniture in indoor spaces, which are then represented as a series of polygons. To validate our method, we compare its performance against ground truth data, demonstrating high accuracy in both simple and complex environments. The core of our study is a comprehensive experiment that evaluates the effectiveness of three evolutionary nature-inspired genetic and three metaheuristic iterative optimization algorithms in solving the sensor placement problem in a complex environment scenario. We perform a statistical analysis to understand the impact of the choice of optimization algorithm and the number of sensors on the achieved spatial coverage. This analysis provides insights into the comparative effectiveness of various evolutionary algorithms in enhancing sensor network design within intricate indoor spaces. In particular, the Artificial Bee Colony algorithm consistently delivered superior results.

Index Terms—Sensor Placement, Evolutionary Algorithms, Point Clouds, Environment Modelling, Sensor Networks

I. INTRODUCTION

With the ever-growing integration of smart systems into our daily lives, the need for a comprehensive understanding of indoor spaces complements the task of finding the optimal deployment of sensors in the field. Modeling residential, commercial, and industrial settings and placing sensors within these environment models offers a cost-efficient and safe way to optimize configurations before implementing complex sensor networks in the real world [1], [2].

These optimized sensor networks can then be deployed in a variety of applications. For instance, networks of cameras may be strategically deployed in diverse indoor environments, including commercial establishments and public spaces, to conduct comprehensive area surveillance, thereby augmenting security measures. Well-positioned temperature and humidity sensors contribute to cost-effective temperature regulation and minimize energy consumption. In retail environments, sensor placement at entrances and throughout the store provides valuable insights into customer behavior and the effectiveness of promotional layouts, which contributes to making informed marketing decisions. In more ways than one, well-designed systems lead to improved user experience.

The first aspect of testing and optimizing sensor networks involves defining and creating a model for the environment in question. Detailed scans of the space are necessary to construct a realistic environment model and compare the result with the ground truth. These scans can be acquired through LiDAR devices or depth cameras, with 3D point clouds emerging as the most common data format. Creating detailed indoor environment models from 3D point clouds is a process that usually involves segmenting the point cloud to identify structural elements like walls, floors, and ceilings and afterward extending to furniture detection and modeling as an additional step [3]. General methods for extracting environment models from 3D scans primarily focus on achieving the most realistic representation of architectural complexities. However, the specific goals of intended applications should guide the development process to ensure the resulting models have practical utility and meet the necessary requirements [4].

The second facet in this context is the sensor positioning problem. It is essential to establish a meaningful quantitative measure for evaluating the effectiveness of a specific sensor network configuration. A key performance metric is network coverage [5] which is directly influenced by the placement of individual sensors and can be defined in several ways. The most prominent variations include area coverage, measuring the ratio of the covered sensor area to the total target area [6], and k-coverage, ensuring each location is covered by at least k sensors [7]. This problem is based on the famous art gallery problem [8] where the goal is to find the smallest number of guards needed to guard an art gallery. Optimizing sensor placement requires a trade-off between achieving good coverage of the area and reducing resource usage.

Placing a single sensor involves adjusting multiple parameters, including its location and orientation while considering factors such as its field of view. Each placement decision impacts the final result, and the difficulty of this problem increases significantly with each added sensor. Given the complexity of the problem, a variety of approaches can be taken. It’s crucial that these strategies are adaptable and enable optimizing for different objectives like maximizing coverage and minimizing costs. This is why various evolutionary, iterative, heuristic, and hybrid strategies can be considered. Opting for established metaheuristic methods allows the exploration of vast solution spaces in search of an optimal outcome. Evolutionary algorithms excel in this regard. To assess and better understand their effectiveness and limitations and choose the most suitable approach, a comparison of these methods is
essential.

In this manuscript, we undertake environment modeling from 3D point clouds and the optimization of sensor placement in complex spaces. Extending on a previous work [9], we propose a comprehensive framework that addresses both challenges, providing an integrated exploration.

A. Related Work

The earliest research aiming to automatically create indoor environment models from point clouds focused on separating the main architectural structures by utilizing plane sweeping for segmentation [10] or employing RANSAC and model-fitting [11]. Expanding on the early RANSAC-based methodologies, authors in [12] proposed a scene representation with cuboids, and then iteratively extracted and optimized these shapes. An alternative approach identified indoor walls by projecting all points onto a floor plane and using distribution histograms with the Hough transform [13], [14]. This field has since been enriched with methods for room segmentation [15]–[20], opening detection [18], [19], [21], and reconstruction of curved elements [22], [23] to gain more accurate space models. More recent approaches use machine learning methods to address more intricate edge cases in extracting a floor plan of an indoor environment from 3D scans with fewer restrictions and assumptions regarding the shape of the space [17], [24], [25].

In the context of Architecture, Engineering, and Construction (AEC), furniture is often perceived as clutter and discarded early in the analysis. However, studies driven by specific use cases, including our investigation on sensor positioning, incorporate furniture into their indoor environment model because they must consider realistic obstacles. Techniques such as primitive fitting [26] and deep learning [27]–[30] play a pivotal role in identifying and categorizing furniture within point clouds. Recognizing common objects is sometimes facilitated by exploiting the abundance of repeated objects in indoor settings [31], [32] or matching them with objects stored in databases [29], [33]. A notable early work [34] introduced an interactive approach to semantic indoor modeling, allowing user-guided segmentation for improved results. Recent studies have focused on analyzing objects with more intricate geometric structures [35], [36], aiming to enhance accuracy in assessing 3D shapes and parameters. Our model focuses on recognizing cube-shaped furniture. While we’re exploring improvements for diverse shapes, this currently meets our needs.

In solving the sensor placement problem, we draw parallels between our problem and the art gallery problem. In the original art gallery problem, guards could monitor a full 360-degree range around their fixed positions within a polygon-shaped art gallery. When calculating space coverage, it’s crucial to consider the sensor’s detection ability [37]. The detection capability of a sensor refers to whether it can “see” an object in the case of binary coverage [38]–[48], or how effectively it can “see” it in the case of probabilistic coverage, which provides a more refined model [49]–[51]. Besides the detection capability, the shape of the area covered by the sensor is also important. The assumption of isotropic sensing ability only applies to certain types of sensors like Bluetooth beacons and wireless beacons, while others, such as cameras and ultrasonic sensors, have directional sensing ability.

Another aspect of the problem is related to the dimension of the target environment. Spaces are often simplified to the level of a floor plan, or two-dimensional boundaries [38], [39], [41], [42], [45]–[48], [52]. To accurately represent a space and derive precise solutions, three-dimensional models are employed. These models consider how the height of obstacles impacts the field of visibility from specific points, albeit incurring additional computational costs. Past research has approximated three-dimensional problems using fixed sensor height and limiting the search space to one plane [44], [50], [53]. It is important to consider the topography of the environment and the obstacles that obscure the sensory area [54]. Solutions may also differ depending on the shape of the space, for example, concave and convex polygons, orthogonal polygons, and polygons with gaps (the gaps in this case correspond to obstacles such as furniture or architectural elements in a closed space). This research focuses primarily on studying environments where the shape and the arrangement of elements conform to an orthogonal grid-like pattern.

Previous studies have employed a variety of optimization algorithms to enhance the efficiency of sensor deployment based on their specific requirements. Techniques such as Integer Linear Programming, Binary Integer Programming, Greedy Search, and random arrangement can be found in the earliest works aiming to place binary directional [38] and omnidirectional sensors [41] in two-dimensional spaces. Efficient methods for determining optimal placement of binary sensors in two dimensions introduced approaches like Individual Particle Optimization [55], Particle Swarm Optimization, Binary Genetic Algorithm, Simulated Annealing [53], Improved Cuckoo Search Algorithm, and Chaotic Flower Pollination Algorithm [6]. Extension to three-dimensional spaces with height maps involved a comparison of deterministic methods with the Covariance Matrix Adaptation Evolution Strategy algorithm, the Limited-memory Broyden-Fletcher-Goldfarb-Shanno method, and Gradient Descent [49], [50], [54]. The Covariance Matrix Adaptation Evolution Strategy proved to give the best results on smaller height maps, while Gradient Descent proved to be better on larger height maps. Additionally, solutions for deploying probabilistic omnidirectional sensors in three dimensions, particularly for drone localization in enclosed environments, have utilized the Gradient Descent method [56].

B. Contributions

The following contributions are presented in this manuscript:

- A novel method for identifying and recording elements within indoor environments from 3D spatial data collected using mobile robots and depth cameras. Our method ensures a detailed and organized representation of a given environment suitable for addressing the problem of optimal sensor placement.
Comparison of three evolutionary nature-inspired genetic algorithms and three metaheuristic iterative optimization algorithms in terms of their effectiveness within the proposed sensor placement framework for two types of sensors applied to obtained space models. We evaluate the impact of chosen sensor placements on our optimization function, utilizing the area coverage metric as a key benchmark.

II. MATERIALS AND METHODS

In this section, we delineate the methodology for data collection, describe the novel method for generating environment models from the acquired data, and assess their accuracy in comparison to the ground truth. The sensor placement procedure is outlined, specifying the employed sensor models, optimization functions, and algorithms, applied to the aforementioned environment models. Additionally, we provide a detailed description of the experimental setup.

A. Data Collection

A synthetic dataset, comprising several simple uncluttered indoor spaces, was created to enable thorough evaluation in a variety of scenarios. The dataset was complemented with a publicly available map named TurtleBot3 House as an example of a more intricate setting. The point clouds that were used in this research were acquired by controlling a mobile robot, traversing the entire space within the virtual environment, and reading the data obtained from a stereo camera. In this stage, the TurtleBot3 Burger robot was utilized, equipped with an Intel RealSense D435i RGBD camera. To facilitate the implementation, the Robot Operating System (ROS) framework was leveraged and coupled with the Gazebo simulator.

When working with stereo cameras and similar 3D imaging devices, preprocessing is often necessary because of the limitations of the underlying technology. One example of such limitations is the decreasing of the field of view as the distance from the sensor increases, which is usually mitigated by moving the 3D acquisition device and retrieving multiple point clouds at different viewpoints and then reconstructing the scene [57]. However, the point cloud coordinates in each frame are tied to the camera’s local system, requiring transformation to a global system to prevent overlaps in the final point cloud. There are two approaches to address this issue, known as point cloud registration or scan matching – one that first finds and then matches pairs of significant points [58], and the other that simultaneously identifies and applies necessary affine transformations [59], [60]. Here, the robot’s odometry, i.e. its displacement from the starting point, is known, simplifying the process. Instead of iterative matching, converting local to global point coordinates involves matrix multiplication using transformation matrices derived from the odometry.

Upon concluding the data acquisition stage, the foundation is laid for the subsequent phase of this study – building representative models from the collected point clouds.

B. Environment Modeling

The first contribution of this paper is a novel method for defining an abstract environment representation from a given point cloud. The chosen model was adopted from our previous work [9]. The model divides the indoor space into two categories – occupied and free space. This approach offers a convenient way to represent the environment, with positive meshes indicating accessible or unobstructed areas and negative meshes representing occupied areas. The overall shape of the environment and its boundaries, primarily walls and floors, are marked as positive meshes. Furniture and other obstacles are marked as negative meshes.

Our approach makes use of the aforementioned model and aims at capturing the depth of three-dimensional spaces while
maintaining the computational efficiency associated with two-dimensional representations. This is achieved by first creating a floor plan that serves as the foundational blueprint of the environment, and then leveraging the available information on the heights of various elements within the space to layer the floor plan with a third dimension.

To offer a thorough understanding of the proposed method, we provide a detailed breakdown of its steps, depicted in Fig. 1.

The initial step focuses on downsampling the point cloud by subdividing it into smaller units called voxels. All points within a voxel are approximated by their centroid, ensuring a precise representation of the data. This process reduces the overall amount of data while retaining crucial features, and therefore plays a key role in enhancing the algorithm’s performance in subsequent stages.

Next, the boundaries of the indoor space and obstacles are identified through RANSAC segmentation [61]. The iterative use of a plane model with a specified normal vector was chosen because RANSAC cannot directly support a cuboid model, which can be approximated by combining multiple planes.

The segmentation process starts with a single plane perpendicular to the z axis, representing the ground plane, followed by alternating segmentation of planes perpendicular to the x and y axes. As new planes are detected, the corresponding points are removed from the original point cloud so that they do not interfere with upcoming segmentation iterations. Segmentation is terminated when all points from the initial point cloud have been successfully classified.

In order to generate the two-dimensional model, essentially composed of polygons, it is necessary to obtain all edges defining the borders of the space elements. Each vertex results in the intersection of three planes – one perpendicular to the x axis (Π_i), one perpendicular to the y axis (Π_j), and one perpendicular to the z axis, representing the ground plane (Π_0). The ground plane Π_0 is common for all intersections, and this process computes the intersection of all Π_i planes with all Π_j planes. The problem of finding the intersection of three planes with known plane coefficients is solved analytically as a system of three equations with three unknowns:

\[ A_0x + B_0y + C_0z + D_0 = 0, \]  \( i \)  
\[ A_i x + B_i y + C_i z + D_i = 0, \]  \( j \)  
\[ A_j x + B_j y + C_j z + D_j = 0, \]  \( k \)

in matrix notation:
\[
\begin{bmatrix}
A_0 & B_0 & C_0 \\
A_i & B_i & C_i \\
A_j & B_j & C_j
\end{bmatrix}
\begin{bmatrix}
x \\
y \\
z
\end{bmatrix} =
\begin{bmatrix}
-D_0 \\
-D_i \\
-D_j
\end{bmatrix}.
\]

Since the equation for the ground plane Π_0 is the same in all systems, the z coordinate should be nearly identical for all points. This simplifies future steps as it allows data to be transferred from the three-dimensional to the two-dimensional domain.

The unordered set of points obtained in this way is not sufficient to uniquely define a list of edges that form polygons. It is first necessary to determine orthogonally adjacent pairs of points.

Then, for each such pair of points, the segmented point cloud is examined to ascertain if any points exist on that edge that are not part of the ground plane. If such points do not exist, that edge aligns with the scenario represented by the dashed lines in Fig. 2, and it is discarded. However, if non-ground points are found, that edge is considered a boundary of some element present in the space and it is added to the list of edges in a polygon. The lowest and highest z coordinates of all the points laying on that edge are recorded, which will be essential when defining the three-dimensional model.

![Fig. 2. The importance of making the right choice when selecting edges](image)

Given the obtained list of edges, a greedy polygonization approach is used for the creation of polygons that represent space boundaries and obstacles. An edge is initially selected from the list, designating one of its endpoints as the polygon’s starting point. The algorithm proceeds to search for the subsequent edge in the list that includes the other endpoint, steadily expanding the polygon until a connecting edge returns it to the initial point. This process is repeated iteratively until all polygons within the space have been successfully constructed.

Data from all edges is used to identify the lowest and highest points for each element. For each polygon, the smallest and largest z coordinates from its edges are retained, indicating the object’s minimum and maximum heights in the three-dimensional model.

The final step in constructing the environment model is to differentiate between space boundaries and obstacles from the collection of polygons obtained in the previous step. Polygons entirely contained within or covered by a larger polygon are classified as obstacles and are marked as negative meshes, while those outlining the outer space boundaries are marked as positive meshes.

For larger environments, this process should be repeated for each room separately, and the results should subsequently be merged, following the divide-and-conquer strategy.

1) Evaluation Metrics: The accuracy of the presented algorithm for constructing an environment model from a point cloud is determined by evaluating how closely the algorithm’s
output aligns with the ground truth, a process closely related to the task of polygon matching [62]. In this examination, two standard metrics are used — Intersection over Union (IoU) and the Dice Similarity Coefficient (DSC). While these metrics originally measure the similarity between two sets, they are adaptable and extendable to computer vision tasks, where they can assess the accuracy of pixel-wise or region-wise segmentation [63]. In this context, IoU quantifies the degree of overlap between the reference polygon $A$ and the generated polygon $\bar{A}$ by measuring the ratio of their intersecting area to their combined area:

$$\text{IoU}(A, \bar{A}) = \frac{|A \cap \bar{A}|}{|A \cup \bar{A}|}.$$  \hspace{1cm} (5)

In contrast, DSC evaluates their similarity by considering their intersecting area relative to their total size:

$$\text{DSC}(A, \bar{A}) = \frac{2|A \cap \bar{A}|}{|A| + |\bar{A}|}.$$  \hspace{1cm} (6)

Both metrics provide insight into how accurately the algorithm has identified and outlined specific regions or objects in the space, with values ranging from 0 (no overlap) to 1 (perfect agreement). However, IoU tends to penalize discrepancies between polygons more strictly and therefore suits worst-case scenarios, while DSC presents a more balanced view and better fits average-case scenarios. By utilizing both metrics, a comprehensive evaluation approach is achieved.

C. Sensor Placement

In our previous work [9], we introduced novel sensor models and an optimization function essential for evaluating sensor placement in a given environment. The proposed sensor models define visibility calculations based on probabilistic coverage models for isotropic and directional sensors. Additionally, the optimization function employs a continuous, single-objective approach to minimize the global loss value, representing the complement of area coverage. In this section, we provide a concise overview of these models and functions, elucidating their role in enhancing the efficiency of sensor placement within a specified environment.

1) Sensor Models: Sensor models are designed to calculate the visibility for voxels surrounding the sensor. When using the probabilistic coverage model, two main types of sensors are considered: isotropic and directional. The visibility function $v_{\text{vs}}$ indicates how efficiently sensor $s$ "sees" voxel $v$. The visibility function is a combination of distance ($v_d$), azimuth ($v_{\phi}$), and inclination ($v_\theta$) and is expressed as:

$$v_{\text{vs}}(v_i, s_j) = v_d(v_i, s_j) \cdot v_{\phi}(v_i, s_j) \cdot v_\theta(v_i, s_j).$$  \hspace{1cm} (7)

These visibilities depend on the sensor type. For isotropic sensors, visibility is constant regardless of direction:

$$v_d(v_i, s_j) = 1, \quad v_{\phi}(v_i, s_j) = 1.$$  \hspace{1cm} (8)

In this study, a model of an isotropic radio beacon is used, with distance visibility $v_d$ based on the Received Signal Strength Indicator (RSSI). The SNR is normalized using $\text{rssi}_{\text{max}}$, resulting in $v_d$. In addition, a model of a Stereo Camera, a directional sensor, is used. The inclination visibility $v_\theta$ is binary, while azimuth visibility $v_{\phi}$ depends on the distance from the image borders. The visibility is influenced by the field of view and stereo-matching algorithm. Refer to our previous work for details [9].

2) Optimization Function: To find optimal sensor positions, an optimization function is necessary, connecting the environment and sensor models based on the chosen metric. The function is a minimization problem, suitable for derivative-free, nonlinear, and constrained optimization algorithms.

The chosen metric is the area coverage ratio, representing the covered area by sensors over the total target area. The optimization aims to minimize the global loss value, the complement of coverage.

The loss $l_{\text{vs}}$ for each sensor-voxel pair is the complement of their visibility:

$$l_{\text{vs}}(v_i, s_j) = 1 - v_{\text{vs}}(v_i, s_j).$$  \hspace{1cm} (9)

As one voxel may be visible from multiple sensors, the voxel loss $L_v$ is the product of losses from each of the $m$ sensors in the set $S$:

$$L_v(v_i, S) = \prod_{j=1}^{m} l_{\text{vs}}(v_i, s_j).$$  \hspace{1cm} (10)

The global loss value $L$ is the average of individual voxel losses $L_v$ for all $n$ voxels in the environment $V$:

$$L(V, S) = \frac{1}{n} \sum_{i=1}^{n} L_v(v_i, S).$$  \hspace{1cm} (11)

Sensor positions, yaw, and pitch angles are defined with continuous values, while voxel positions use discrete values derived from the environment rasterization parameter.

3) Optimization Algorithms: As mentioned earlier, the determination of sensor placement using the described optimization functions can leverage any optimization algorithm capable of nonlinear constrained optimization without the need for function derivations. In this study, six optimization algorithms are specifically considered and compared.

The first three proposed algorithms belong to the category of population-based metaheuristic genetic optimization algorithms. The algorithms, such as Particle Swarm Optimization (PSO) [64], Artificial Bee Colony (ABC) [65] and Fireworks Algorithm (FWA) [66], [67], are based on population optimization and swarm intelligence concepts. All three algorithms were used and compared for mobile robot path planning [68], solving sensor placement problems [9], [69], as well as for a variety of other problems [70]–[72].

In addition, there are three metaheuristic iterative algorithms in which the future state (sensor placement) at each step depends only on the placement of the current iteration. These algorithms, including the Nelder-Mead Method (NMM) [73], the Hill Climbing Algorithm (HCA) [74] and Simulated Annealing (SA) [75], [76], have been used and compared in various applications [69], [77]–[79].

Due to the nature of problems with unknown solutions, optimization cannot be performed towards a fixed target.
Therefore, the "fixed-cost" approach is used, where the number of executions of optimization functions is limited for each algorithm in a given test case [80]. The number of executions is set to $E = 200 \cdot D$, where $D$ is the number of independent variables.

4) Experimental Setup: To investigate area coverage challenges, an experiment was conducted using the TurtleBot3 House indoor space model, which covers an area of $86.68m^2$ and includes six distinct rooms with several obstacles. This model, chosen for its complexity, allows testing sensor placement in a real-world scenario, providing more valuable insights into the optimization methods compared to simpler hypothetical inorganic spaces used in our previous work. However, this comes with the trade-off of higher computing resource usage and longer execution times.

The experiment examined the impact of optimization algorithm selection on the relative area coverage. The number of sensors assumed to proportionally affect coverage and execution time, served as the second independent variable. The space was discretized into $0.2 \times 0.2m$ voxels. We employed up to 50 isotropic Bluetooth Low Energy sensors and, separately, up to 20 directional Stereo Camera sensors. When combined with six optimization algorithms, this setup resulted in a total of 420 diverse test cases. To mitigate stochastic variability in optimization algorithms, each test case was repeated 30 times. The computational demands of this extensive experimentation were met through the utilization of the University of Rijeka’s "Bura" supercomputer.

III. RESULTS AND DISCUSSION

In this section, we present the outcomes of our study, beginning with the quantitative evaluation and qualitative analysis of the acquired environment models. Subsequently, we present the results of the sensor placement task within the TurtleBot3 House environment, differentiating between isotropic and directional sensors. Finally, we analyze and interpret the obtained results, substantiated with appropriate statistical analysis.

A. Environment Modeling

Table I shows the summary of the quantitative results for our proposed method for obtaining a 2D model of the space from a 3D point cloud following the IoU and DSC metrics. Better results are obtained for smaller spaces with simpler geometric structures which are easier to model accurately.

<table>
<thead>
<tr>
<th>Environment</th>
<th>env1</th>
<th>env2</th>
<th>TurtleBot3 House</th>
</tr>
</thead>
<tbody>
<tr>
<td>IoU</td>
<td>0.987</td>
<td>0.982</td>
<td>0.923</td>
</tr>
<tr>
<td>DSC</td>
<td>0.994</td>
<td>0.991</td>
<td>0.960</td>
</tr>
</tbody>
</table>

Qualitative results are presented in Fig. 3. While our algorithm effectively captures the fundamental spatial structure, it occasionally encounters challenges in accurately delineating furniture, particularly evident in the case of the three tables within the TurtleBot3 House. This limitation primarily stems from the sparsity of data points along the sides of the tables in the point cloud representation, posing difficulties for RANSAC in correctly segmenting the corresponding planes. Consequently, this issue propagates errors in subsequent processing stages.

![Fig. 3. Comparison of the ground truth (blue) and the results obtained from the environment modeling algorithm (red).](image1)

B. Sensor Placement

Within this section, the results of the sensor placement task are presented. Results show maximum area coverage achieved for isotropic Bluetooth Low Energy sensors, and for directional Stereo Camera sensors.

1) Isotropic Sensors: Table II provides an overview of the area coverage achieved by the selected optimization algorithms across test cases involving Bluetooth Low Energy sensors. Notably, ABC and PSO algorithms achieve the best results, which suggests that these algorithms are particularly effective in addressing the sensor placement problem when optimizing sensor locations to achieve maximum area coverage.

In order to statistically analyze the effects of the optimization algorithm used and the number of sensors on the average coverage, we performed a two-way ANOVA. Namely, a $6 \times 50$ ANOVA was utilized, with Algorithm (6 instances)
TABLE II
AREA COVERAGE STATISTICS FOR ALL TEST CASES CONTAINING 1 TO 50
BLUETOOTH LOW ENERGY SENSORS USING SIX OPTIMIZATION
ALGORITHMS.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>ABC</th>
<th>PSO</th>
<th>HCA</th>
<th>FWA</th>
<th>SA</th>
<th>NMM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average</td>
<td>0.543</td>
<td>0.517</td>
<td>0.450</td>
<td>0.424</td>
<td>0.416</td>
<td>0.402</td>
</tr>
<tr>
<td>St. dev.</td>
<td>0.224</td>
<td>0.212</td>
<td>0.180</td>
<td>0.173</td>
<td>0.180</td>
<td>0.179</td>
</tr>
<tr>
<td>Max value</td>
<td>0.819</td>
<td>0.777</td>
<td>0.679</td>
<td>0.656</td>
<td>0.659</td>
<td>0.649</td>
</tr>
</tbody>
</table>

and Number of sensors (1 to 50) being the within-subjects factors. The test yielded the following results:

- Mean coverage differed statistically significantly between optimization algorithms:

\[ F(5, 8700) = 27901.422, p < .001. \]

Post-hoc analysis with a Bonferroni adjustment confirmed that ABC statistically achieves the best results among all algorithms.

- Mean coverage differed statistically significantly between a number of sensors:

\[ F(49, 8700) = 36194.408, p < .001. \]

Post-hoc analysis with a Bonferroni adjustment confirmed a statistically significant difference in the average coverage achieved with each added isotropic sensor. Notably, this effect diminishes when using more than 40 sensors.

- The interaction between the factors Algorithm * Number of sensors is statistically significant:

\[ F(245, 8700) = 82.680, p < .001. \]

Fig. 4 depicts how optimization algorithms compare in their effectiveness at maximizing area coverage and reinforces the conclusion that ABC and PSO consistently outperform other optimization algorithms across various sensor configurations. As expected, the sequential progression of the line graphs corresponds to the systematic increase in the number of sensors. In addition, Fig. 5 offers a qualitative perspective, illustrating exemplar scenarios of isotropic sensor placement across the space.

2) Directional Sensors: Table III offers a summary of the area coverage attained by the chosen optimization algorithms in scenarios featuring Stereo Camera sensors. This time, the ABC algorithm emerges as significantly superior to other algorithms, showcasing its effectiveness in addressing the sensor placement problem specific to Stereo Cameras.

TABLE III
AREA COVERAGE STATISTICS FOR ALL TEST CASES CONTAINING 1 TO 20
STEREO CAMERA SENSORS USING SIX OPTIMIZATION ALGORITHMS.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>ABC</th>
<th>PSO</th>
<th>HCA</th>
<th>FWA</th>
<th>SA</th>
<th>NMM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average</td>
<td>0.650</td>
<td>0.443</td>
<td>0.423</td>
<td>0.380</td>
<td>0.352</td>
<td>0.313</td>
</tr>
<tr>
<td>St. dev.</td>
<td>0.251</td>
<td>0.156</td>
<td>0.144</td>
<td>0.142</td>
<td>0.142</td>
<td>0.131</td>
</tr>
<tr>
<td>Max value</td>
<td>0.933</td>
<td>0.640</td>
<td>0.610</td>
<td>0.569</td>
<td>0.537</td>
<td>0.482</td>
</tr>
</tbody>
</table>

In order to statistically analyze the effects of the optimization algorithm used and the number of sensors on the average coverage, we performed another \( 6 \times 20 \) two-way ANOVA, with Algorithm (6 instances) and Number of sensors (1 to
20) being the within-subjects factors. The test yielded the following results:

- Mean coverage differed statistically significantly between optimization algorithms:
  \[ F(5, 3480) = 11417.835, p < .001. \]
  Post-hoc analysis with a Bonferroni adjustment confirmed that ABC statistically achieves the best results among all algorithms.
- Mean coverage differed statistically significantly between a number of sensors:
  \[ F(19, 3480) = 6249.471, p < .001. \]
  Post-hoc analysis with a Bonferroni adjustment confirmed a statistically significant difference in the average coverage achieved with each added directional sensor.
- The interaction between the factors \textit{Algorithm} * \textit{Number of sensors} is statistically significant:
  \[ F(95, 3480) = 85.222, p < .001. \]

Fig. 6 provides a visual representation of the comparative performance of the different optimization methods. The ABC optimization algorithm demonstrates even more notable capability when applied to Stereo Camera sensors, outperforming its counterparts in achieving superior area coverage. This enhanced performance suggests that the algorithm’s inherent ability to explore the solution space and converge on optimal configurations is particularly well-suited for addressing the complexities posed by this specific sensor type. Again, Fig. 7 illustrates examples of directional sensor placement across the space.

Our environment modeling method is able to accurately represent spatial boundaries, primarily architectural structures such as walls and floors, and obstacles that impact the field of visibility. Additionally, it can extrapolate the heights of objects, effectively transforming the initial two-dimensional representation into a comprehensive three-dimensional model. The evaluation metrics yield highly favorable results for spaces characterized by an orthogonal grid, evident in both simpler and more complex indoor environments.

The proposed method for determining optimal sensor placement uses a loss function based on the overall area coverage provided by each sensor across all voxels in a given space. In our experiment, we employed six stochastic optimization algorithms to address the sensor placement problem in a test space that mirrors real indoor living and working spaces in terms of shape, size, and content. The results of this study underscore the significance of algorithm selection in maximizing the performance of sensor networks, considering the nuanced compatibility between algorithmic strategies and sensor types. In particular, the ability of the ABC algorithm to adapt and explore the search space effectively makes it an appealing choice for sensor placement in our scenario. The significantly higher area coverage value achieved by the ABC algorithm shows its superiority in the exploration phase. This becomes clear in the example with the directional sensors, where the addition of the yaw angle strongly influences the performance of the five other optimization algorithms.

Future work includes the improvement of the proposed method for indoor spaces with more complex architectural structures such as stairs and curved surfaces, spaces that don’t conform to the Manhattan assumption and identification of openings. As an additional step, we plan to assess the model in a real-world context using a physical robot. Our sensor positioning method could incorporate reinforcement learning or evaluate different types of coverage, especially \( k \)-coverage.
REFERENCES


