B. Urtasun, I. Andonegui and E. Gorostegui-Colinas, "Sparse sampling-based view planning for complex geometries," in *IEEE Sensors Journal*.

DOI: https://doi.org/10.1109/JSEN.2024.3372622

AUTHOR QUERIES

AUTHOR PLEASE ANSWER ALL QUERIES

PLEASE NOTE: We cannot accept new source files as corrections for your article. If possible, please annotate the PDF proof we have sent you with your corrections and upload it via the Author Gateway. Alternatively, you may send us your corrections in list format. You may also upload revised graphics via the Author Gateway.

Carefully check the page proofs (and coordinate with all authors); additional changes or updates WILL NOT be accepted after the article is published online/print in its final form. Please check author names and affiliations, funding, as well as the overall article for any errors prior to sending in your author proof corrections.

AQ:1 = Please confirm whether the representation of the author name "Benat Urtasun" throughout the article is correct.

AQ:2 = Please confirm or add details for any funding or financial support for the research of this article.

AQ:3 = Please confirm whether the edits made in the current affiliation of all the authors are correct.

AQ:4 = The in-text citations of the figures are out of order and are not sequential. Figs. 1 and 2 should be cited before Fig. 3 and Fig. 12 should be cited before Fig. 13. Please update in-text citations so that all figures are cited in sequential order.

AQ:5 = Please provide the expansions for the acronyms GRASP, ORI, POS, IKFAST, FCL, VTK, OMPL, and GOM.

AQ:6 = Please provide the appropriate section number for the phrase "following sections."

AQ:7 = Please provide the publisher location for Ref. [38].

AQ:8 = Please provide the year of completion when the author Imanol Andonegui received the B.Sc., M.Sc., and Ph.D. degrees.

AQ:9 = Please confirm whether the hierarchy of the degree names for the author Imanol Andonegui is correct.

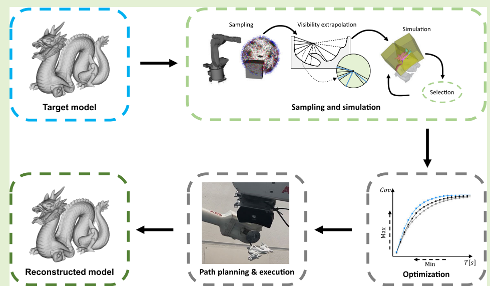
AQ:10 = Please specify the university name for the B.Sc. degree of the author Imanol Andonegui.

AQ:11 = Please provide the location (city and country) for University of the Basque Country, University of Sydney, University of Navarra (San Sebastian), and CEIT.

Sparse Sampling-Based View Planning for Complex Geometries

Benat Urtasun, Imanol Andonegui, and Eider Gorostegui-Colinas

***Abstract*—In this article, an automatic sampling-based view planning algorithm is proposed, for accurate 3-D reconstruction of complex geometry parts present in manufacturing. The initial viewpoint sampling method is able to lower the complexity of the algorithm by creating a sparse visibility bipartite graph relating the targeted surface patches, with the potential viewpoints [camera poses defined in SE(3)], which are contained in the surroundings of the object. This graph is used to sample and simulate a subset of viewpoints, employing an iterative greedy parallel set cover which weights the coverage of the sparse and simulated visibility. This method prematurely rejects poor candidates and prioritizes the viewpoints providing an increased coverage, with no expensive preprocessing of the 3-D models. A randomized Greedy heuristic with local search has been proposed to maximize the coverage, while minimizing the total inspection time, first with the set cover of the simulated viewpoints, and second with the sequencing of the viewpoints and robot positioning with obstacle avoidance. Furthermore, the performance of the system is demonstrated on a set of complex benchmark models from the Stanford and MIT repositories, yielding a higher coverage with a lower computational runtime compared with existing sampling-based methods. The validation of the full system has been carried scanning a Stanford Dragon positioned with a 12-axis kinematic chain composed of two robots.**



***Index Terms*—Cameras, clusterization, combinatorics, Greedy, metaheuristics, optimization, robotics, sensor deployment, smart sensors, surface reconstruction, traveling salesman problem (TSP), view planning.**

I. Introduction

A. Motivation

AUTOMATED inspections have gained significance within the smart manufacturing context as they are necessary for many downstream applications or quality assurance. These systems are commonly required to inspect a surface that will ensure the fulfillement of the required specifications. Usually, the complete coverage of the surface of interest requires a set of capture from different viewpoints. The associated camera network design or the automation of the robotic inspection can be a lengthy process with many delays. The automatic resolution of this aspect is called a view planning problem (VPP), which revolves on the maximization of the coverage of the surface to be inspected while simultaneously reducing the total inspection time. Considering that increased coverage benefits from a higher number of capture points and minimizing the time involves its reduction, the simultaneous optimization of both objectives is not trivial. This work addresses this problem with contributions (Section I-D) that enable the minimization of computation and execution time of the inspection, facilitating the inspection of complex geometries in a reduced time.

B. Related Works

Typically, the solution to the VPP for an unknown 3-D object is handled with a next best view (NBV) approach. This method determines iteratively the subsequent position that will reveal the greatest possible portion of the component's surface or its immediate environment for the robot. Some methods recur to octomaps which chart the surroundings of the occupied, empty, and unknown space, to estimate a probabilistic map of the information gain [1], enabling the determination of the upcoming pose. Even if this strategy is useful for reverse engineering and path finding of robots [2], [3], [4], among other applications, it requires an intermittent online capture and processing, artificially extending the process and

Manuscript received 15 December 2023; revised 21 February 2024; accepted 24 February 2024. The associate editor coordinating the review of this article and approving it for publication was Prof. Xiaofeng Yuan. *(Corresponding author: Benat Urtasun.)*

Benat Urtasun is with the LORTEK Technological Centre, Basque Research and Technology Alliance (BRTA), 20240 Ordizia, Spain, and also with the Robotics and Automation Group, Electronic and Computer Science Department, Faculty of Engineering, Mondragon University, 20500 Mondragón, Spain (e-mail: benaturtasun@gmail.com).

Imanol Andonegui is with the Robotics and Automation Group, Electronic and Computer Science Department, Faculty of Engineering, Mondragon University, 20500 Mondragón, Spain.

Eider Gorostegui-Colinas is with the LORTEK Technological Centre, Basque Research and Technology Alliance (BRTA), 20240 Ordizia, Spain.

Digital Object Identifier 10.1109/JSEN.2024.3372622

incurring in other inefficiencies. Approaching the VPP with an approximate model that enables the simulation of the inspection allows the usage of different heuristics and methods to attain a predictable result.

Depending on the final goal, many specification criteria have been utilized. For instance, in a surface reconstruction problem the minimum sampling density and variance of the point clouds are considered [5], and in a network placement problem, the main objective is to make a complete coverage of the scene [6] with the minimum number of viewpoints.

The classical sampling-based VPP, which employs an approximate model of the targeted surface, such as the one exposed by Scott [5], starts with the sampling of viewpoints, its subsequent simulation, and the final set cover ensuring the maximum coverage. The sampling of the viewpoints starts by decimating [7] or resampling [8] the surface mesh, which yields another mesh with a different distribution and density of the primitives. This mesh is used to sample the surface points by selecting the vertices or the barycenters of the mesh primitives. These points are used to sample a set of a priori ideal viewpoints with a normal incidence angle from a distance corresponding to the maximum optical resolution, which is defined as the center of the depth of field (DOF), as described in Algorithm 1.

Algorithm 1 Sample Offset DOF [5]

function SAMPLEOFFSETDOF($Mesh, z_f, z_n, n_{cams}$)
  $Mesh' \leftarrow ResampleMesh(Mesh, n_{cams})$
  $P, N \leftarrow SampleBarycenters(Mesh', n_{cams})$
  $Cams \leftarrow \emptyset$
  for each $p_i \in P$ **do**
    $o_i \leftarrow p_i + n_i(z_f + z_n)/2$
    $Cams \leftarrow Cams \cup ToFrame(o_i, -n_i)$
  return $Cams$

Other viewpoint sampling methods such as the one exposed by Jing et al. [9], summarized in Algorithm 2, generate a volume surrounding the object, computed by calculating the perpendicular at the surface points of the object, and adding the minimum and maximum distance of the DOF. This 3-D volume is used to randomly sample the origins of the viewpoints, and their orientations are determined with a potential function of the neighboring surface normals.

Algorithm 2 Sample Potential Field [9]

function SAMPLEPOTENTIALFIELD($Mesh, z_f, z_n, n_{cams}$)
  $Mesh' \leftarrow ResampleMesh(Mesh, n_{cams})$
  $V \leftarrow dilate(Mesh', z_f) - dilate(Mesh', z_n)$
  $O_{cams} \leftarrow RandomSampling(V, n_{cams})$
  $Cams \leftarrow \emptyset$
  for each $o_i \in O_{cams}$ **do**
    $v_i \leftarrow potentialField(o_i)$
    $Cams \leftarrow Cams \cup ToFrame(o_i, v)$
  return $Cams$

The resulting set of viewpoints is then simulated considering the visibility, as well as the incident angle θ, as illustrated in Fig. 3(a), among other factors, resulting in a visibility vector of the surface points for each viewpoint, $\overrightarrow{A_i}$. The visibility of the N viewpoints, regarding M surface points conforms a visibility matrix, $\mathbf{A_{vis}} = (\overrightarrow{A_1}, \ldots, \overrightarrow{A_N})$, which can be interpreted as a bipartite graph relating both disjoint sets, as formulated by Tarbox and Gottschlich [10]. This data structure, which can be interpreted as a bipartite graph, enables a combinatorial formulation of the VPP as a set cover problem (SCP), to maximize the coverage of the surface with the minimum number of viewpoints.

Considering that the total area to cover is finite, the likelihood of visualizing the same surface patches increases as the number of viewpoints rises. The diminishing returns of this problem is one aspect of its submodularity associated with the total overlap of the visibility [11]. Therefore, the coverage and number of viewpoints are two conflicting objectives which must be approximated in a reasonable time scale. The optimization of the problem has been previously solved employing well-established metaheuristics such as, greedy [12], linear programming [13], Lagrangian relaxation [14], simulated annealing [15], particle swarm optimization [16], and genetic algorithms [17].

The conventional greedy set cover [12], described in Algorithm 3, repeatedly selects the next column (viewpoint) of $\mathbf{A_{vis}}$, which maximizes the coverage of the remaining uncovered points, until the whole set is covered in $O(\log n)$, [18]. Its unweighted cost, as well as the deterministic selection criteria, precludes the exploration of alternative solutions, which can be improved with a randomized selection [19]. Another aspect to consider is that its parallelization is able to reduce the runtime with a similar solution, so long the problem is subdivided into buckets of maximal near-independent sets [20]. The set cover yields a set of unordered inspection frames which might be used to position static cameras or generate an inspection trajectory, minimizing the inspection time and considering the kinematic constraints of the robot and camera attached to the robot wrist, by employing a combinatorial optimization known as the traveling salesman problem (TSP).

Algorithm 3 Greedy Set Cover

function GREEDYSETCOVER($A = \{A_1, \ldots, A_n\}$)
  $Sol \leftarrow \emptyset$
  while $|Uncovered(Sol)| > 0$ **do**
    Select j that maximizes $|A_j \cap Uncovered(Sol)|$
    $Sol \leftarrow Sol \cup j$
  return Sol

One of the main drawbacks of all these systems is that they do not use complex geometries instances in the exposed results, as well as a typical runtime to solve the problem on the order of minutes [5], [16], [21], [22].

Considering that the simulation of the viewpoints takes a significant share of the total runtime of this problem, the sampling of an optimal subset of viewpoints is an important aspect of the problem. Most of the conventional viewpoint sampling methods are able to restrict its sampling space, but they do not take into account any information from the surrounding geometry, which limits their ability to extrapolate the mutual visibility of the viewpoints. The occlusion ratio of a point should a priori correlate to the number of incident cameras in a visibility matrix, but it does not retain any spatial information to prioritize the sampling of viewpoints associated with complex surface patches. Some pseudoillumination models, employed in 3-D rendering to shade the surfaces, such as ambient occlusion [23], map a scalar field in the surface, by computing the ratio of occluded local random rays. This yields a scalar field associated with the vertices or faces of the model, with high values related to concave regions, internal geometries, or high curvature regions. However, this mapping of the surface is nevertheless unable to determine the best location of the viewpoints for each surface patch.

All the mentioned studies expose different methods to solve the problem, but they typically involve an expensive mesh preprocessing which is prone to alter the original surface and its topology, introducing defects such as normal inversion affecting the visibility and accuracy of the simulation. Another factor to take into account is the extended computational times exposed by these studies, which impose restrictions on the scale and complexity of the inspected part. Furthermore, the minimization of the inspection time focuses mainly on the SCP without considering the sequencing of the viewpoints restricted by the axes of the robot positioning the sensor and its workspace. The contributions addressing these shortcomings are enumerated in Section I-D.

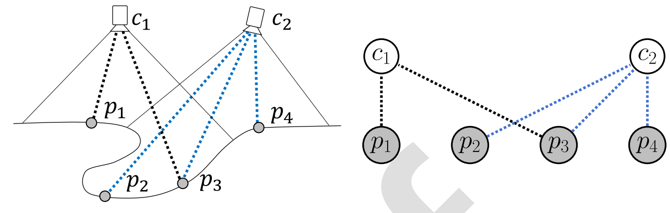
C. Problem Formulation

The VPP consists on the determination of a minimum set of scanning viewpoints $\mathbf{C_p}$ to cover a surface. The surface of the inspected part, S is composed of a set of vertices in $\mathbb{R}^3$, and a collection of polygons, which are defined as an adjacency list of vertices. Another aspect to consider is that the set of viewpoints must be contained in a space belonging to the special Euclidean group SE(3) [24] and surrounding S. The coverage of S by $\mathbf{C_p}$ must also fulfill a set of specification parameters γ, which have been defined in this article as: 1) the minimum density, defined as the maximum distance between the points, $\delta_{\mathbf{max}}[m]$ and 2) the maximum incident angle of the camera toward a point, noted as $\theta_{\mathbf{max}}$.

The combinatorial approach of the VPP requires the discretization of both S and $\mathbf{V_c}$ (space of possible camera poses), yielding a set of M points or polygons $\mathbf{P} = \{p_1, \ldots, p_M\}$, and N viewpoints, $\mathbf{C} = \{c_1, \ldots c_N\}$ with $\mathbf{C_p} \in \mathbf{C}$. The determination of the visibility of a point p_i, regarding a viewpoint c_j, can be formulated as a binary scalar (0—nonvisible and 1—visible), a_{ij} that takes into consideration the direct line of sight and the specification compliance. Therefore, the computation of the visibility of a viewpoint viewpoint c_j, regarding the whole set of points $\mathbf{P}$, can be defined as a binary visibility vector, $\overrightarrow{A_j} = (a_{1j}, \ldots, a_{Mj})^T$, with a_{ij} being the visibility of p_i regarding c_j. The combination of all the viewpoint visibility vectors conforms a binary visibility matrix [10], with the points and the viewpoints corresponding to the rows and columns, respectively, noted as $\mathbf{A_{vis}} = (\overrightarrow{A_1}, \ldots, \overrightarrow{A_N})_{|\mathbf{P}|\times|\mathbf{C}|}$.

Note that $\mathbf{A_{vis}}$ can be represented as a bipartite graph of two disjoint sets, $\mathbf{P}$ and $\mathbf{C}$. Fig. 1 shows their symbolic relation in (a), as well as its bipartite graph in (b), with the vertices on the top symbolizing the viewpoints, the points below, as well as the edges representing their visibility. The visibility matrix of this figure is shown as follows.

Consequently, we can define the VPP as the joint minimization of (1) the number of viewpoints $|\mathbf{C_p}|$ with $\mathbf{C_p} \in \mathbf{V_c}$



(a) (b)

Fig. 1. Visibility as a bipartite graph. (a) Symbolic representation of the visibility with two cameras covering a surface discretized in four points and the dotted line showing the visibility of each point toward the cameras. (b) Bipartite visibility graph corresponding to the left side in this figure.

and (2) the ratio of uncovered points of $\mathbf{P}$, subjected to the visibility and specification compliance γ as follows:

$$\min_{\mathbf{C_p}\in\mathbf{V_c}} \left(f(\mathbf{C_p}), |\mathbf{C_p}|\right) \quad \text{with } f(\mathbf{C_p}) = 1 - \frac{1}{M}\sum_{i}^{M} \bigcup_{j}^{N} \overrightarrow{A_j}. \tag{1}$$

Note that $f(\mathbf{C_p})$ represents the ratio of uncovered points considering the union of the visibility vectors of $\mathbf{C_p}$.

D. Contributions

A sampling-based view-planning system is exposed with a set of distinct contributions aimed at reducing the runtime of the VPP and the total inspection time of the robot.

1) A novel sampling view-planning that employs a sparse representation of the underlying visibility, reducing the sampling space with a clusterization preserving the relation between the space of the viewpoints and the surface.
2) A sampling and simulation algorithm that does not require any expensive preprocessing of the 3-D model, yielding typical runtimes close to 1 s.
3) An improved greedy heuristic for the SCP and robot traveling salesman (rTSP) problem, with a randomized local search, analogous to GRASP [19], to minimize the time to traverse the viewpoints by the robot.
4) Results validated with a set of 20 complex benchmark models demonstrating a higher coverage with a lower runtime compared to existing sampling-based methods, as well as the evaluation of the full system scanning a Stanford Dragon with two robots.

II. Proposed Method

Based on the submodular property of the VPP [11], a set of assumptions can be established to approximate the underlying visibility matrix, which can be used for efficient sampling of the simulated viewpoints.

Taking into account that this is a sampling-based view-planning, the proposed method estimates a visibility matrix which serves as the basis for the optimization of the objectives to attain the maximum coverage and minimum inspection time. An overview of the system is displayed in Fig. 2, starting by sampling the surface (Section II-A), which does not require an expensive pre-processing of the mesh. A subsequent estimation of the visibility yields a sparse visibility matrix

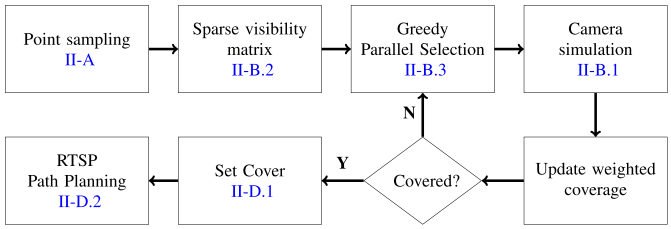


Fig. 2. System overview.

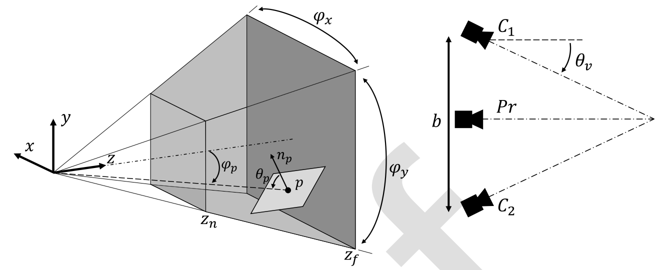


Fig. 3. Visibility evaluation. (a) Pinhole view-frustum with a DOF between z_n and z_f, FOV with φ_x and φ_y. A ray directed from the focal point toward p with an incident angle θ_p is drawn with a dotted line. (b) Stereo camera and projector relative position with a baseline b and vergence angle θ_v.

(Section II-B2), which is employed to iteratively select a set of viewpoints (Section II-B3), weighing both the sparse and simulated visibility (Section II-B1), taking into account the accessibility of the robot (Section II-C). The resulting set of viewpoint vectors links **P** on a dense visibility matrix, which serves as the basis for the minimization of the total inspection time (Section II-D), first by reducing the set of viewpoints that ensures the coverage by employing a Greedy randomized SCP (Section II-D1) and a subsequent reordering of the viewpoints, taking into account the robot (Section II-D2), in a problem known as the RTSP.

A. Surface Point Sampling

As previously stated, depending on the specification parameters of resolution and inherent variable sampling density of most surface reconstruction algorithms employed in the generation of the 3-D models, it is necessary to produce a uniform point sampling of the surface, S. In this system, a modified version of the algorithm exposed by Corsini et al. [25], has been implemented, starting with a Monte Carlo point sampling of the surface with a higher resolution of the predefined $\delta_{\mathbf{max}}$, typically by a factor of 10. A subsequent subsampling is carried out by iteratively selecting random points and discarding the neighboring ones at $\delta_{\mathbf{max}}$ radius. The neighboring points are typically selected, employing spatial indexers, such as kd-trees [26], or hash tables [27], among others methods. The iterative selection terminates when the projected number of points, based on the area is reached, or no points remain on the uncovered list.

B. Visibility Calculation

The determination of the visibility in this scenario starts by the determination of the sparse visibility matrix and the subsequent iterative selection of viewpoints and camera simulation. Note that in this scenario, Section II-B1 is exposed before Section II-B2 to present the view frustum.

1) Camera Simulation Employed 3-D Camera: The employed scanner in this work is a precalibrated Gocator 3520, composed of two 5-MP cameras and a 100-W blue light fringe projector, allowing for the 3-D measurement, so long the projector has the co-visibility of one camera, enabling the reduction of the shadows and mutual occlusions present in complex geometries. It is based on a structured light phase-shifting scanner, projecting a set of shifted sinusoidal patterns, which ultimately allows the pixelwise association between the cameras and the projector. This enables the triangulation of the scanned surface points, taking into account the calibrated optics and their relative positions, as illustrated in Fig. 3(b).

TABLE I
GOCATOR 3520 VIEW-FRUSTUM PARAMETERS

$\varphi_\mathbf{x}$	$\varphi_\mathbf{y}$	$\mathbf{z_n}$	$\mathbf{z_f}$	$\mathbf{R_x}$	$\mathbf{R_y}$	$\mathbf{b}$	$\theta_\mathbf{v}$	$\delta_\mathbf{min}$
30º	40º	280mm	430mm	1944	2592	180mm	14º	0.08mm

As a result, a conservative visibility evaluation of the scanner fuses the visibility of each device as a combination of the visibility of the projector and the cameras. Therefore, the visibility of a point p is defined as $v = v_{\text{proj}} \cap (v_{c1} \cup v_{c2})$ with v_{proj}, v_{c1} and v_{c2}, being the separated visibility of the projector and both cameras respectively.

The visibility of each device toward the surface points has been assessed individually through a three-step process. First, by examining the view-frustum containment of each point [28]; second, by evaluating specification compliance; and finally, by ensuring an unobstructed line of sight.

A pinhole model has been used to describe the view-frustum of each camera, as well as the projector. Fig. 3(a) displays the view-frustum as a truncated pyramid in a darker shade, with φ_x and φ_y being the field of view (FOV) constrained by the sensor rectangular shape in the horizontal and vertical axes, respectively. The minimum optical resolution is ensured by constraining the DOF, between z_n and z_f. The relative position of the stereo camera with the projector is shown in Fig. 3(b), being θ_v, the vergence angle in the XZ plane and b, the distance between the cameras. Table I depicts the parameters associated with the Gocator 3520, assuming the same view-frustum for the three devices, with R_x and R_y being their resolution.

Note that the maximum incidence angle depends on the reflectance of the surface, the exposure, and aperture among other factors which has been determined empirically, yielding a value of $\theta_{\mathbf{max}} = 70°$.

The specification compliance of the minimum resolution, $\delta_{\mathbf{max}}$, has been estimated with a similar approach to the one exposed by Scott [5], which can be approximated with the following equation:

$$\delta_p = \frac{R_p \Delta\varphi}{H\left(\theta_p < \theta_{\mathbf{max}}\right) \cos\theta_p}. \quad (2)$$

where $R_p = z_p/(\cos\varphi_p)$ is the distance between p and the focal point, $\Delta\varphi = \min((\varphi_x/R_x), (\varphi_y/R_y))$ is the minimum angular resolution of the sensor, $H(\theta_p < \theta_{\mathbf{max}})$ being the Heaviside step function with $\theta_{\mathbf{max}}$ being the maximum incidence angle, and $(\cos\theta_p)^{-1}$ modeling the Lambertian reflectance associated with the incidence θ_p, as shown in Fig. 3(a).

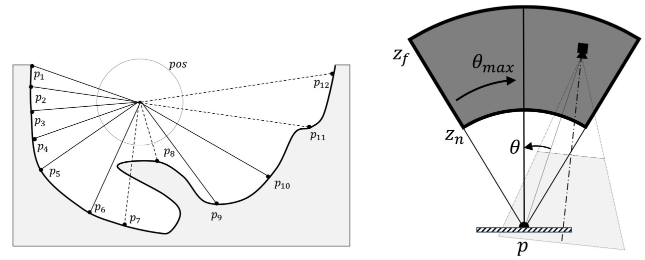
Another aspect to consider is the computation of the direct line of sight of the cameras, which is known to be a complex problem [29], which can limit the scale and complexity of the VPP. The two main ways to solve this problem consist of the ray casting of the optical rays originating from the sensor to the scene, and alternatively the projection of the world into the plane of the sensor.

Using the ray casting to estimate the visibility implies the evaluation of the intersection between each ray with all the geometric primitives of the scene. The alternative, based on the Z-buffer method [30] has an exponential decay [31] in its precision, and the rasterization of the projection implies that the framebuffer resolution must be sufficiently small to visualize the specified surface resolution, $\delta_{\mathbf{max}}$.

In this article, a ray-tracing technique, such as Embree [32], has been integrated to project rays from the camera toward the remaining points within the view-frustum. This process adheres to specification compliance and effectively separates the visibility runtime from the sensor's resolution.

2) Sparse Visibility Matrix: The sparse visibility matrix is based on the extrapolation of the visibility of the neighboring viewpoints. The visibility from a point pos, surrounding the surface is illustrated in Fig. 4(a) showing the visible surface points with solid rays, which are restricted by the direct line of sight, DOF, and their respective incident angle. Therefore, if two of the remaining rays are contained in the FOV of a viewpoint, both of their respective surface points will be visible. For instance in Fig. 4(a), a 45° FOV camera with its optical axis aligned with the ray of p_1 will also visualize p_2. The same idea can be extended for the viewpoints located on an Euclidean radius around pos. The sparse visibility matrix can be defined as an approximation of the dense visibility matrix described in Section I-C; however, it exhibits two clear differences. The first one lies in the fact that it relates the visibility toward a random subset of $\mathbf{P}$ denoted by $\mathbf{P}_{\text{sp}}$. The second one is that it has an explicit partition of the viewpoints. This is due to the way the visibility is extrapolated with a spatial indexation of the viewpoints, as it will be explained later. Therefore, the sparse visibility matrix can be denoted as follows: $\mathbf{A}_{\text{sp}} = (\mathbf{A_1}, \ldots, \mathbf{A_n})$, where $\mathbf{A}_{i_{|\mathbf{P}_{\text{sp}}|\times|\mathbf{C_i}|}}$ is the submatrix of the extrapolated visibility of a subset of viewpoints $\mathbf{C_i}$, regarding $\mathbf{P}_{\text{sp}}$. The sparse visibility matrix is built based on the efficient extrapolation of the local visibility, starting with the sampling of a collection of viewpoint axes from each surface point, and the subsequent extrapolation of the visibility.

a) Point visibility sampling: The first phase involves sampling a set of optical axes associated with the points on the surface with a direct visibility. The process starts by selecting a random fraction κ of $\mathbf{P}$, denoted by $\mathbf{P}_{\text{sp}}$. For each point p in $\mathbf{P}_{\text{sp}}$, a subset of fixed vectors is sampled, representing the optical axes of potential viewpoints directed to p. To ensure the visibility of an optical axis k_p toward p with its normal n_p, a point visibility space is defined with two equations depending on the pinhole parameters of the camera and k_p: 1) $z_n \leq k_p^T n_p \leq z_f$ and 2) $(k_p/(|k_p|))^T n_p > \cos\theta_{\mathbf{max}}$, representing DOF containment and feasible angle of incidence. This volume has the shape of a partial spherical cone, with its vertex and axis coincidental to the point p and surface normal, n_p, respectively. The maximum and minimum radii correspond to the DOF range, and the cone half-angle is associated with the maximum incidence angle, $\theta_{\mathbf{max}}$, as illustrated in Fig. 4(b). A set of vectors pointing to p is sampled from this volume with a 3-D uniform grid and a Δd resolution. The direct line of sight is evaluated by ray casting from k_p toward p, discarding the occluded ones. Based on the experiments, the following grid sampling resolution gives good results:



(a) (b)

Fig. 4. Camera sampling. (a) Symbolic representation of the omnidirectional visibility from a point in space pos, casting rays to the visible points in solid lines conditioned by the distance, incident angle, and the occlusions. (b) Point visibility sampling volume, representing a partial spherical cone, with its vertex and axis coincidental to p and surface normal, n, respectively.

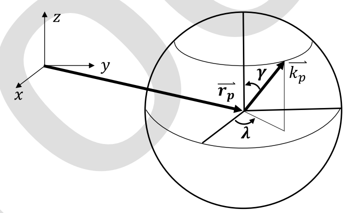


Fig. 5. Optical axes grid parameters in $\mathbb{R}^3$ for r_p and the latitude γ and longitude λ of k_p regarding the frame of the object.

$$\Delta d = \frac{1}{3}\left(\frac{z_f + z_n}{2}(\tan\varphi_x + \tan\varphi_y) + z_f - z_n\right). \quad (3)$$

Δd represents an average of the DOF, and the dimensions corresponding to the mid-plane cross section of the viewfrustum.

b) Visibility extrapolation: The second phase consists of the extrapolation of the visibility of the neighboring optical axes. Considering that each optical axis is linked to a surface, the extrapolation has been carried out in two steps. The first one consisting of the binning of the optical axes employing a grid which partitions the Euclidean space $\mathbb{R}^3$, and the orientation space with spherical coordinates, as shown in Fig. 5.

The grid is built by indexing the optical axes, assigning five integer scalars (three for position and two for orientation) to each optical axis, which are then sorted first by the Euclidean position, and subsequently by the orientation. This effectively groups the optical axes belonging to the same orientation bin, denoted by ORI, contained on an Euclidean bin, denoted by POS. As a resul the consecutive elements with the same

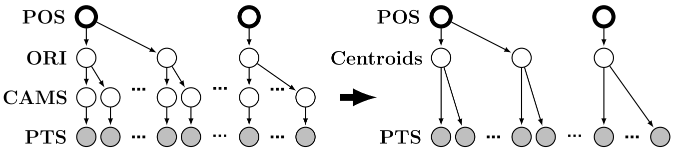


Fig. 6. Hierarchical binning is depicted with a two-level spatial indexing of the optical axes, with an Euclidean POS, and orientation ORI partitioning, corresponding to the first and second levels, respectively. The left side of the figure shows that each orientation bin contains a set of optical axes which are linked to a single point each. The right side displays the centroids of the axes of each bin linked to all incident points of ORI.

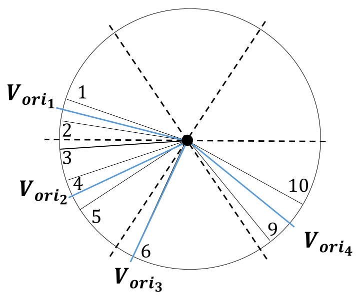


Fig. 7. Rays directed to the points illustrated in Fig. 4(a) from pos partitioned in 60° bins with the axes centroids of each bin in blue.

orientation belong to the same bin. The left side of Fig. 6 displays the relation of the ordered optical axes, denoted by CAMS, contained in the orientation and position bins. So long, the Euclidean and angular resolution of the grid, Δd and $\Delta\beta$, respectively, are sufficiently small, all the optical axes contained in the same orientation and position, bin will have similar r_p and k_p vectors, resulting in a comparable visibility. Therefore, the centroid of the optical axes of each orientation bin inherits the predominant visibility of the bin. The right side of Fig. 6 displays the centroids of the orientation bin inheriting the visibility of the surface points from the optical axes. Experiments have shown that the Euclidean resolution of the grid Δd, described in (3) gives good results, as well as the following angular resolution: $\Delta\beta = \min(\varphi_x, \varphi_y)/4$. The centroid of the optical axes is determined as follows: $r_C = (1/n)\sum_{i=0}^{n} r_i$ and $k_C = (\sum_{i=1}^{n} k_i)/(|\sum_{i=1}^{n} k_i|)$.

Note that the ordered list of points of the spatial binning and the strides of the orientation bins associated with the clustered camera centroids can be seamlessly copied to the row and column index buffers of a binary compressed row sparse (CRS) matrix, respectively. The resulting CRS matrix conforms an approximation of $\mathbf{A_{vis}}$ with a lower density. Considering that the hierarchical binning groups the camera centroids by Euclidean bins, the sparse visibility matrix can be as noted as a set of n column blocks corresponding to the Euclidean bins POS, denoted by $\mathbf{A}'_{\text{sp}} = (\mathbf{A_1}, \ldots, \mathbf{A_n})$.

One of the drawbacks of the binning is that the resulting clusterization depends on the origin of the spatial partition. For instance, a cluster of optical axis can be divided, resulting in two contiguous centroids, instead of one that clusters the group. Fig. 7 shows a set of outgoing rays from pos directed to the points shown Fig. 4(b), with an angular partition of $\Delta\beta = 60°$, represented with dotted lines, and their respective centroids drawn in blue.

In this example, the viewpoint aligned with V_{ori_1} will probably see most of the points visualized by V_{ori_2}, but none of the points corresponding to V_{ori_3}. Alternatively, V_{ori_2}, will probably visualize most of their adjacent ones. This redundant co-visibility of the axis centroids can be used to further increase the number of edges in the sparse bipartite graph. Therefore, the co-visibility of the local axes centroids contained in an Euclidean bin, $K_{\text{bin}} = \{k_1, \ldots, k_m\}$, can be formulated as a symmetric adjacency matrix, denoted by: $\mathbf{A}_{\text{cams}} = (\ldots e_{ij} \ldots)_{m\times m}$, with $e_{ij} = k_i^T \cdot k_j > \cos\Delta\beta$. The extrapolation of the visibility within the Euclidean bin, has been carried out with a graph composition of the sparse visibility matrix, $\mathbf{A}_{\text{sp}}$ and the optical axis orientation adjacency matrix $\mathbf{A}_{\text{cams}}$, with the following binary matrix multiplication, $\mathbf{A}_{\text{sp}} = \mathbf{A}'_{\text{sp}} \times \mathbf{A}_{\text{cams}}$, with $\mathbf{A}_{\text{sp}}$ being the final sparse visibility matrix. Fig. 8 shows a visibility graph corresponding to Fig. 4(a), with the upper row corresponding to a set of viewpoint nodes and their mutual adjacency represented by the dotted edges. As a result, the nodes in the bottom are associated with the points $\mathbf{P}$, which are connected to the viewpoints $\mathbf{C}$, either by the initial binning with gray edges or by the subsequent extrapolation in black.

The following expression shows the graph composition of the visibility extrapolation illustrated in Fig. 8, corresponding to Fig. 4(a):

$$\underset{A'_{\text{sp}}}{\begin{pmatrix}1&0&0&0\\1&0&0&0\\0&1&0&0\\0&1&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&1\\0&0&0&1\end{pmatrix}} \times \underset{A_{\text{cams}}}{\begin{pmatrix}1&1&0&0\\1&1&1&0\\0&1&1&0\\0&0&0&1\end{pmatrix}} = \underset{A_{\text{sp}}}{\begin{pmatrix}1&\mathbf{1}&0&0\\1&\mathbf{1}&0&0\\\mathbf{1}&1&\mathbf{1}&0\\\mathbf{1}&1&\mathbf{1}&0\\\mathbf{1}&1&\mathbf{1}&0\\0&\mathbf{1}&1&0\\0&0&0&1\\0&0&0&1\end{pmatrix}}. \tag{4}$$

The generation of the sparse visibility is summarized in Algorithm 4.

Algorithm 4 Build Sparse Visibility

1: **function** BUILDSPARSEVISIBILITY($\mathbf{P}$, $\theta_{\max}$, $CampPars$, κ)
2: $\mathbf{P_{sp}} \leftarrow$ SubsamplePoints($\mathbf{P}$, κ)
▷ Sample and ray-cast optical axes for each point (Section II-B2a)
3: $\mathbf{Axes} \leftarrow$ PointVisibility($\mathbf{P_{sp}}$, $\theta_{\max}$, $CamPars$)
4: $\mathbf{Centroids}$, $\mathbf{A_{sp}}$, $Bins \leftarrow$ VisibilityExtrapolation($\mathbf{P_{sp}}$, $\mathbf{Axes}$)
▷ Optical axes centroids to viewpoints
5: $\mathbf{C} \leftarrow ToFrames(\mathbf{Centroids})$
▷ Filter invalid Robot viewpoints
6: $\mathbf{C} \leftarrow$ FilterInvalidViewpoints($\mathbf{C}$)
7: **return** $\mathbf{P_{sp}}$, $\mathbf{C}$, $\mathbf{A_{sp}}$

Note that the viewpoints are calculated from the centroids with a random rotation of the z-axis in line 5 of Algorithm 4.

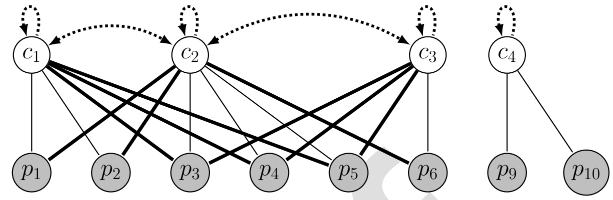


Fig. 8. Bipartite graph relating the visibility of the viewpoints on top and the points at the bottom related to Fig. 4(a). The gray edges are associated with the binning, and the black ones to the extrapolation. The dotted lines denote the orientation adjacency of the viewpoints.

3) Greedy Iterative Selection: The sampling and simulation of the viewpoints are based on a greedy set cover (alg. 3), weighting the coverage globally, with a local parallel selection. The selection penalizes the number of covers of each point by weighting both the extrapolated visibility ($\mathbf{A}_{\text{sp}}$), and the simulated viewpoints, up to a minimum number of covers, $\min_{\text{cov}}$. Considering that the neighboring viewpoints, contained in the same Euclidean bin (POS), have a higher overlap of the surface visibility, compared with the farthest ones, it enables its parallel selection in buckets of maximal near-independent sets [20], approximating the sequential greedy set cover solution with a shorter runtime. The proposed method to sample and simulate the viewpoints is exposed in Algorithm 5.

Algorithm 5 Sparse Iterative Sampling

1: **function** SPARSEITERATIVESAMPLING($\mathbf{P}$, min_{cov}, CamPars)
▷ Initialize $\mathbf{P}$, Centroids and Sparse visibility, alg. 4
2: $\mathbf{P_{sp}}$, $\mathbf{C}$, $\mathbf{A_{sp}}$ ← BuildSparseVisibility($\mathbf{P}$, $\theta_{\max}$, $CampPars$, κ)
▷ Initialize camera viewpoints, visibility matrix, and coverage vector
3: $Cams \leftarrow \emptyset$, $\mathbf{A_{vis}} \leftarrow \emptyset$ $\overrightarrow{Cov} \leftarrow \emptyset$
▷ Iterative selection and camera simulation
4: **while** True **do**
▷ Weighted uncovered points vector
5: $\overrightarrow{Uncov} \leftarrow \max(0, 1 - \frac{\overrightarrow{Cov}}{min_{cov}})$
6: $Cams' \leftarrow \emptyset$
▷ **Greedy parallel selection**
7: **for each** $\mathbf{A} \in \mathbf{A_{sp}}$ **do**
▷ Remaining weighted coverage
8: $\overrightarrow{UncovCams} \leftarrow A^T \times \overrightarrow{Uncov}$
9: Select i maximum row of $\overrightarrow{UncovCams}$
10: **if** $\overrightarrow{UncovCams}_i \geq 1$ **then**
11: Discard i^{th} camera in A
12: $Cams' \leftarrow Cams' \cup \mathbf{C}_i$
13: **if** $Cams' == \emptyset$ **then**
14: **break**
15: $\mathbf{A}'_{\mathbf{vis}}$ ←CameraSimulation($\mathbf{P}$, $Cams'$, CamPars)
▷ Save viewpoints and simulated visibility
16: $Cams \leftarrow Cams \cup Cams'$, $\mathbf{A_{vis}} \leftarrow \mathbf{A_{vis}} \cup \mathbf{A}'_{\mathbf{vis}}$
17: Add the dense and sparse coverage of $\mathbf{P_{sp}}$ to $\overrightarrow{Cov}$
18: **return** $Cams$, $\mathbf{A_{vis}}$

After calculating the sparse visibility matrix with Algorithm 4 in line 2, the vector $\overrightarrow{\text{Cov}}$, which counts the accumulated covers of each point of $\mathbf{P}_{\text{sp}}$ is initialized, as well as the final set of viewpoints, Cams and the simulated visibility matrix $\mathbf{A}_{\text{vis}}$ of $\mathbf{P}$. The iterative selection starts by initializing the vector $\overrightarrow{\text{Uncov}}$, which negatively weights the accumulated covers of each point of $\mathbf{P}_{\text{sp}}$, up to a minimum number of covers, $\min_{\text{cov}}$, as shown in line 5. The parallel selection within each bin POS, starts by calculating the weighted new coverage $\overrightarrow{\text{UncovCams}}$, of each viewpoint in line 8, with $\mathbf{A}$, being the block of $\mathbf{A}_{\text{sp}}$ corresponding to the viewpoints contained in POS. Afterward, the viewpoint with the maximum value, greater or equal to one, is saved. The parallel selection, yields at most a viewpoint for each bin, which is then simulated in line 15 and saved in $\mathbf{A}_{\text{vis}}$. The accumulated coverage of the points $\overrightarrow{\text{Cov}}$, is updated with the summation of the dense and sparse visibility of Cams$'$. This process is repeated until no viewpoints are selected.

C. Robot Accessibility Testing

The accessibility of the viewpoints is evaluated based on the existence of a valid robot configuration with no collision. A fast inverse kinematic (IK) solver, such as IKFAST [33], has been employed returning, the complete set of solutions. The sampling of robot configurations for kinematic chains with more than six degrees of freedom has been carried out with two different methods. In the case of the external positioning axis, a random or uniform sampling for each redundant axis is sufficient, and for multiple robotic arms, a Cartesian bounding box is defined to randomly sample the possible configurations, as described in the results.

The resulting robot configurations are subsequently tested for any intersection of the robot with the scene. The collision detection is typically handled using a two-phase approach consisting of an initial broad phase and a subsequent narrow phase. The broad phase employs a simplified primitive geometry of the objects to discard the evaluation of distant objects. Some implementations use the sort and sweep algorithm to evaluate the overlap of the projected bounds of the primitives into the three axes. While other approaches recur to a parallel spatial cell subdivisions to evaluate the collision of objects contained in the same cell. The second phase computes the exact contact points of the intersected geometry. A collision detection library, such as FCL [34] has been implemented in this instance with both the broad and narrow phases.

D. Inspection Time Optimization

After simulating the visibility of the sampled viewpoints, the problem must be able to minimize the total inspection time, maximizing the coverage. The joint optimization of both problems is notoriously hard which has motivated the division of the problem in two steps. The first one consists of the minimization of the number of selected viewpoints on an SCP, analogous to the greedy set cover Alogirthm 3. And, a second phase aims at minimizing the time to visit each viewpoint by simultaneously reordering them considering the robot configurations, which is a variation of the TSP, known as the RTSP. Since the solution of both problems can be formulated as an ordered list, a sequential greedy insertion [12], can be employed in an iterative manner. Some heuristics, such as GRASP [19], add some randomization to the greedy heuristic by choosing among the k candidates for the solution, instead of the best one. The proposed generic resolution of both problems is displayed in Algorithm 6, consisting of the generation of an initial solution S, conformed as an ordered list. This solution is iteratively optimized, by employing a similar scheme to a variable neighborhood search (VNS) [35], first by discarding g elements and a subsequent randomized insertion of the k nearest neighbors. The resulting solution S' is preserved so long it improves S. The local search is terminated after $t_{\max}$ seconds, or $l_{\max}$ iterations with no improvements. To enable the adaptation of the generic optimization scheme, to the SCP and RTSP, the following functions must be altered accordingly, RandomizedGreedyInsertion, discardGRandom, as well as compareSolution. The adaptation of the proposed scheme is detailed in the following sections. AQ:6

1) SCP: Both the initial solution shown in line 2, as well as the insertion of the local search in line 6 from Algorithm 6, have been implemented with Algorithm 7, considering the previously calculated visibility matrix, $\mathbf{A}_{\text{vis}}$ =

Algorithm 6 Greedy Variable Neighborhood Search

1: **function** RANDOMIZEDGREEDYVNS($\mathbf{A} = \{A_1, \ldots, A_n\}$, k, g)
 ▷ Initial solution
2: $S' \leftarrow$ RandomizedGreedyInsertion($\mathbf{A}$, $\emptyset$, k)
3: $S \leftarrow S'$, $l \leftarrow 1$
4: **while** $l \leq l_{max} \cap t < t_{max}$ **do**
5: $S' \leftarrow$discardGRandom(S', g)
6: $S' \leftarrow$RandomizedGreedyInsertion(A, S', k)
7: **if** $compareSolution(S', S)$ **then** $S \leftarrow S'$, $l \leftarrow 1$
8: **else** $l \leftarrow l + 1$
9: **return** S

$(\overrightarrow{A_1}, \ldots, \overrightarrow{A_N})_{|\mathbf{P}| \times |\mathbf{C}|}$. Note that the insertion of viewpoints stops after reaching a coverage ratio, η_{vis} is reached as shown in line 2. It starts by determining the number of uncovered points of the solution S of each viewpoint, resulting in the vector $\overrightarrow{\text{Covers}}$. Subsequently, a column among the k maximums of $\overrightarrow{\text{Covers}}$ is choosen. The random removal of g elements in the unordered solution S, discardGRandom follows a uniform distribution. The iterative local search saves the solution S', so long it has a lower cardinality regarding the best S, or an improved coverage with the same cardinality.

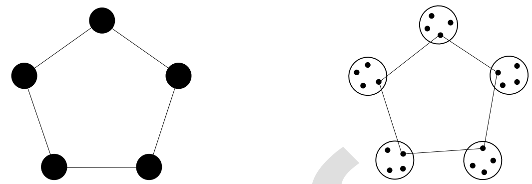
Algorithm 7 Randomized Greedy SCP

1: **function** RANDOMIZEDGREEDYINSERTIONSCP($\mathbf{A_{vis}} = \{A_1, \ldots, A_N\}$, S, k)
2: **while** $\frac{1}{M}|Uncovered(S)| > 1 - \eta_{vis}$ **do**
 ▷ New covers for each viewpoint
3: $\overrightarrow{Covers} \leftarrow (\ldots, |Uncovered(S) \cap A_j|, \ldots)_{j \in \{1,\ldots,N\}}$
4: Pick random j column within the k maximums of $\overrightarrow{Covers}$
5: $S \leftarrow S \cup j$
6: **return** S

2) Robot Traveling Salesman Problem: The minimum set of viewpoints with a coverage ratio of η_{vis} that complies with the specifications must be sequenced to minimize the time to visit each viewpoint. The scanning space, or task space, T, is contained in SE(3), which is associated with the end effector of the robot. The projection of the robot space R, onto T, known as the forward kinematic (FK), is unique, but its opposite, the IK, does not share the same property. Non-holonomic robots, as well as singular points in T, might even have infinite IK solutions. Consequently, every target t_i within the set T forms a cluster of robot configurations denoted as $R_i = \{r_{ij}\}$, thereby extending the TSP to a Clustered TSP (CTSP).

In most industrial inspections, the start of any robot routine coincides with the end on a "home" configuration, r_{home}, conforming a Hamiltonian tour traversing all the viewpoints. The RTSP is a particularization of the CTSP, which in some approximations leverages the duality of the robot and task space to reduce the complexity of the problem [36]. Fig. 9(a) displays the Hamiltonian tour on a TSP graph in the task space, and b represents the corresponding RTSP.

The complete set of clusters, including home, is defined as $\mathbf{A} = \{A_0, \ldots, A_{N-1}\}$, with each cluster A_i composed by a varying number of robot configurations, with $A(i, j) = a_{ij}$, being the robot configuration j of the target i. A tour S is defined as an ordered list of M pairs, $\{x, y\}$, with x and y being the set point number and its associated configuration respectively.



(a) (b)

Fig. 9. Scan sequencing. (a) TSP. (b) Robot-TSP.

The time to transition from a robot configuration $\overrightarrow{a_{ij}}$ to $\overrightarrow{a_{kl}}$ is defined as: $\text{cost}(\overrightarrow{a_{ij}}, \overrightarrow{a_{kl}}) = \max(|\overrightarrow{a_{ij}} - \overrightarrow{a_{kl}}| \oslash \overrightarrow{\omega})$, with $\overrightarrow{\omega}$ being the axes velocities of the robot and $\oslash$ the elementwise vector division. As a result, the cost of a tour S is the summation of all the segment costs. And, the function compareSolution of line 7 in Algorithm 6 for the RTSP determines if S' has a lower cost compared with S.

Adapting the function RandomizedGreedyInsertion for the RTSP has resulted in Algorithm 8, which assigns a random configuration of A when the sequence is empty, and then iteratively chooses the configurations that are among the k minimum costs of the unvisited target configurations.

The implementation of discardGRandom for the RTSP, defined in line 5 from Algorithm 6, erases a set of g contiguous elements of the circular sequence, yielding a unique gap for the subsequent insertions.

Algorithm 8 Randomized Greedy Insertion RTSP

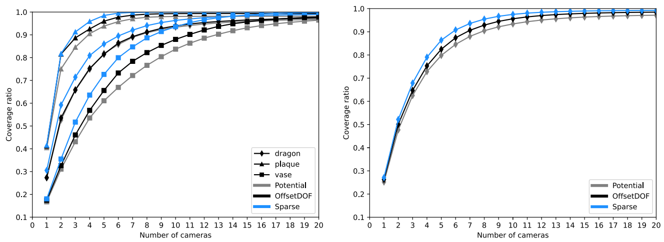
1: **function** RANDOMIZEDGREEDYINSERTIONRTSP($\mathbf{A_{vis}} = \{A_1, \ldots, A_N\}$, S, k)
 ▷ Hamiltonian cycle enables random start
2: **if then**$|S| == \emptyset$
3: pick random $i \in \{0, \ldots, M-1\}$ and $j \in \{0, \ldots, |R_i| - 1\}$
4: $S_0 \leftarrow \{i, j\}$
 ▷ Insert in the first gap, $next$
5: $curr \leftarrow firstBeforeNull(S)$, $next \leftarrow (curr + 1)\%M$
6: **repeat**
 ▷ Costs from S_{curr} to remaining viewpoint configurations
7: $Costs$=$\{\{Cost(S_{curr}, \{i, j\})\}_{\forall i \in (\{0,\ldots,M-1\} - S), \forall j \in \{0,\ldots,|R_i|-1\}}$
8: Pick random $\{i, j\}$ among k minimums in $Costs$
 ▷ Add to sequence
9: $S_{next} \leftarrow \{i, j\}$
10: $curr \leftarrow next$, $next \leftarrow (curr + 1)\%M$
11: **until** $S_{next} \neq 0$
12: **return** S

III. Experiments and Results

The evaluation of the proposed method has been conducted in two phases. The first one compares the view-planning system without the robot. The second phase benchmarks the full system with two robotic arms and a printed Stanford Dragon.

A. Synthetic View Planning

To evaluate the performance of the contributions, regardless of the employed kinematic chain, a set of four models from the Stanford repository and 16, from the MIT CSAIL Textured Models Database has been simulated throughout the pose generation, simulation, and the Greedy Set Cover exposed in Algorithm 3 selecting up to 20 viewpoints. The quantitative evaluation has been carried out by employing the area under



(a) (b)

Fig. 10. Coverage sequence up to 20 viewpoints comparing the proposed method (sparse) and two alternative methods. (a) Comparison between three models of the dataset. (b) Average of the whole dataset.

the curve (AUC) [37], measuring the accumulated information gain of the final Greedy selection sequence.

The minimum resolution is $\delta_{\max} = 0.001$ m with a maximum incidence angle, $\theta_{\max} = 70°$, employing the camera parameters associated with the Gocator3520, as shown in Table I.

Two alternative pose generation methods have been compared, the first one proposed by Scott [5], implemented with Algorithm 1, and a second exposed by Jing et al. [9] following Algorithm 2. Both methods sample a predetermined number of viewpoints based on the resolution and the area of the mesh as: $n_{\text{cams}} = (1/20)(\text{area}_{\text{model}}/\delta^2_{\max})$. Since both methods require a mesh resampling, the method exposed by Schroeder et al. [7] has been used, which is implemented in VTK with the operator vtkDecimatePro [38]. Note that the presented method employs the following parameters: $\kappa = 0.25$ and $\min_{\text{cov}} = 15$.

Table II displays the results of the 20 models and the three methods, reporting the coverage of 2, 4, and 6 viewpoints, as well as the AUC and the runtime in seconds. Note that to reduce the randomness, the results are averaged in ten runs, executed in a laptop with a Ryzen 9 5900HX with 16 parallel threads in eight cores and 32 GB of RAM. Fig. 10(a) illustrates three instances of the coverage sequence, and Fig. 10(b) displays the average of the whole set.

B. Real Tests

1) Setup: The tests have been carried out with a kinematic chain composed of two manipulators with six axes, consisting of an ABB IRB 6700 235/2.65 carrying the scanner and an ABB IRB4600 60/2.05 with a printed Stanford Dragon tied to the 6th axis, as illustrated in Fig. 11. To replicate the real setup in the simulation, the kinematic chain shown in Fig. 11 has been calibrated employing common methods. The FK of both robots, associated with the frames of their flanges regarding their respective bases, ${}^{\text{rob}}T_{\text{FL}}$, have been determined using the nominal DH parameters of both robots. The relative position of their bases, ${}^{\text{rob}_{\text{cam}}}T_{\text{rob}_{\text{part}}}$ has been calibrated following the default method provided by the robot controller with an error of 2.2 mm. As for the hand-eye calibration associated with the relative position of the scanner coordinate system, ${}^{\text{FL}_{\text{cam}}}T_{\text{cam}}$, centered in the projector focal point, regarding the flange of its robot, it has been estimated with the quaternion method [39], with a set of 12 captures employing a checkerboard pattern, yielding a square error of 0.278 mm and 0.012°. The frame of the inspected part regarding the flange of the robot, ${}^{\text{rob}_{\text{part}}}T_{\text{part}}$, has been determined by averaging the registration of the model

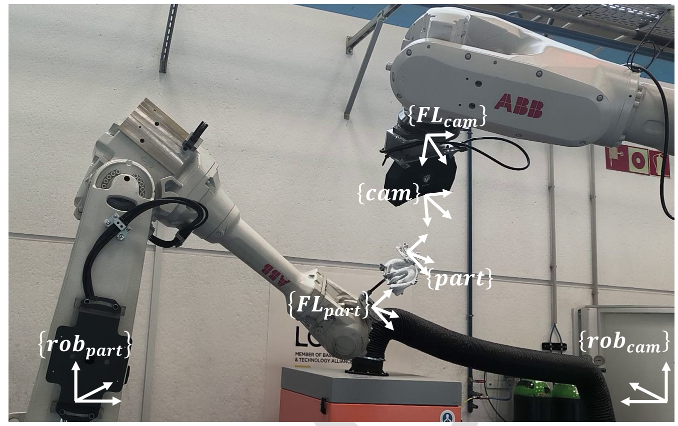


Fig. 11. Setup and approximate frames of the kinematic chain carrying the scanner and the part.

with six captures yielding an average error of 15.67 mm and 0.44°.

2) Reconstruction Analysis: The employed parameters of the system are presented in Table III.

The resulting sampling has simulated 474 poses for a set of 32 671 surface points. The final selection has employed the randomized Greedy SCP with 16 instances in parallel for 10 s, selecting the best solution. ~~Fig. 13(a) shows the comparison of the resulting sequence of the conventional Greedy SCP, as well as the corresponding accumulated visibility of the scanned point clouds.~~ The solution is composed of s[illegible] viewpoints which have been sequenced, employing the RTSP algorithm described in Section II-D2. The 12 axes robot configurations of the capture poses have been sampled, first by selecting a random pose of the viewpoint on a Cartesian bounding box of 0.5 × 0.5 × 0.5 m to determine the corresponding frame of the other robot. The dense path with obstacle avoidance of the resulting sequence of robot configurations has been planned with RRT-Connect [40] implemented in OMPL [41], which has been subsequently post-processed to generate two robot programs compatible with the controller enabling a synchronized execution. The accumulated errors of the kinematic chain alter the resulting pose which provokes a deviation from the simulated visibility. The Cartesian deviation of the robot has been measured by registering the point cloud from the theoretical frame of the model, regarding the model itself. The total overlap of the point clouds has been determined, first by discarding the points that do not attain the minimum resolution, $\delta_{\max}$, determined by a minimum number of neighbors, $\min_{\text{NN}}$, within a radius, $r = 2\delta_{\max}$, employing the following expression: $\min_{\text{NN}} = (\pi r^2)/\delta^2_{\max}$. And, second by estimating the number of points of the simulated point cloud within a $2\delta_{\max}$ distance of the registered capture. ~~Fig. 13(b) shows the registration distance with the resulting overlap~~. The seven captures of the inspection are presented in the columns of Fig. 12, with the top and middle rows displaying the projected point clouds of the simulated and scanned viewpoints. The third row displays the model with the point cloud overlapped to the simulated in red, and the non-overlapping in green, as well as the synthetic points which are not scanned in blue. The surface reconstruction of the model has followed a conventional method consisting of the prealignment of the clouds to the

TABLE II

Comparison on Models From Stanford and MIT Repositories, Reporting the Absolute Coverage of 2, 4, and 6 Cameras, the AUC Up to 20 Cameras and the Total Runtime. All Results Are Averaged With Ten Runs

		Armadillo	Buddha	Bunny	Dragon	Bird	Bowl	Bust	Column	Flowerpot	Gargoyle	Goblet	Head	Lion	Mural	Obelisk	Owl	Plaque	Pot	Thinker	Vase
Potential [9]	C2	0.524	0.484	0.573	0.526	0.585	0.392	0.508	0.444	0.375	0.539	0.376	0.410	0.448	**0.638**	0.337	0.492	0.750	0.293	0.533	0.311
	C4	0.781	0.717	0.853	0.749	0.878	0.651	0.801	0.713	0.601	0.829	0.582	0.717	0.707	0.849	0.582	0.807	0.905	0.507	0.805	0.535
	C6	0.887	0.819	0.960	0.858	0.953	0.819	0.930	0.845	0.740	0.923	0.703	0.881	0.839	0.925	0.742	0.919	0.958	0.633	0.913	0.669
	AUC	0.774	0.719	0.832	0.754	0.836	0.705	0.798	0.730	0.642	0.803	0.610	0.744	0.728	0.828	0.641	0.791	0.871	0.554	0.790	0.585
	T[s]	2.001	6.988	**0.274**	5.120	0.723	**0.910**	2.184	2.454	5.321	1.236	**2.184**	7.015	3.160	2.992	8.712	**0.611**	1.248	10.533	1.047	**0.867**
Scott [5]	C2	0.587	0.487	**0.627**	0.535	0.610	0.488	0.530	0.460	0.390	0.545	0.414	**0.424**	0.488	0.637	0.308	0.494	0.815	0.289	0.566	0.326
	C4	0.851	0.727	0.894	0.752	0.898	0.705	0.822	0.762	0.651	0.844	0.622	**0.749**	0.747	0.800	0.543	0.804	0.926	0.543	0.848	0.569
	C6	0.944	0.833	0.978	0.864	0.975	0.888	0.948	0.927	0.778	0.934	0.764	**0.891**	0.874	0.909	0.706	0.930	0.978	0.712	0.921	0.733
	AUC	0.826	0.728	0.855	0.757	0.851	0.763	0.811	0.778	0.680	0.812	0.659	**0.755**	0.758	0.813	0.614	0.795	0.894	0.615	0.808	0.626
	T[s]	2.088	6.892	0.372	5.708	0.781	1.251	2.279	2.668	6.046	1.335	2.347	7.688	3.373	3.381	9.645	0.847	1.355	11.596	1.150	1.230
Proposed	C2	**0.611**	**0.516**	0.621	**0.592**	**0.642**	**0.501**	**0.571**	**0.510**	**0.401**	**0.572**	**0.426**	0.403	**0.497**	0.624	**0.338**	**0.507**	**0.817**	**0.336**	**0.592**	**0.355**
	C4	**0.864**	**0.784**	**0.908**	**0.809**	**0.950**	**0.751**	**0.865**	**0.843**	**0.703**	**0.854**	**0.657**	0.711	**0.780**	**0.853**	**0.605**	**0.824**	**0.959**	**0.611**	**0.857**	**0.636**
	C6	**0.967**	**0.880**	**0.983**	**0.895**	**0.993**	**0.912**	**0.975**	**0.957**	**0.851**	**0.943**	**0.811**	0.890	**0.899**	**0.957**	**0.778**	**0.939**	**0.995**	**0.818**	**0.941**	**0.800**
	AUC	**0.842**	**0.769**	**0.859**	**0.795**	**0.871**	**0.784**	**0.837**	**0.815**	**0.727**	**0.823**	**0.699**	0.746	**0.776**	**0.838**	**0.664**	**0.804**	**0.908**	**0.685**	**0.826**	**0.680**
	T[s]	**0.328**	**1.198**	0.356	**1.283**	**0.280**	1.510	**0.679**	**0.733**	**3.575**	**0.609**	6.806	**2.074**	**1.188**	**0.811**	**2.214**	0.990	**0.376**	**3.885**	**0.652**	2.770

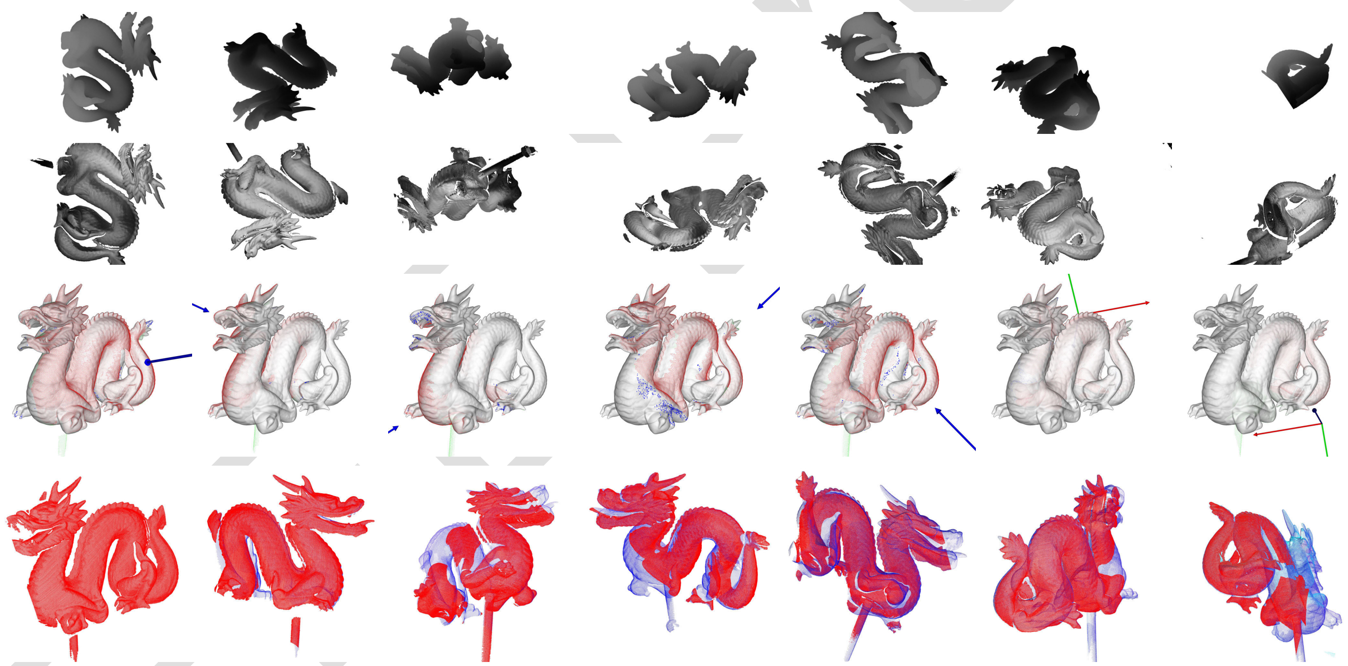


Fig. 12. Comparison of the simulated poses and the resulting point clouds with the first and second rows displaying the projected cloud from the viewpoint of the simulated and scanned pose. The third row displays the resulting cloud registered to the model and its corresponding viewpoint. And, the last row shows the incremental registration of the clouds with the registered cloud in red and the previous ones in blue.

TABLE III

Robot View-Planning Parameters

Specifications		Iterative sparse		SCP				RTSP		
$\delta_{\max}$	$\theta_{\max}$	min_{cov}	κ	η_{vis}	t_{scp}	g	k	t_{rtsp}	g	k
$1mm$	70º	35	0.2	90%	$10s$	2	4	$10s$	2	4

frame of the robot flange carrying the scanned object and a subsequent incremental registration with a modified iterative closest point (ICP) [42]. The ICP has been implemented using the point cloud library [43], employing a different objective function [44], and a correspondence estimation based on a normal shooting coupled with normal rejection. The set of registered clouds is the basis for the surface reconstruction employing the software GOM inspect. Fig. 14 shows the resulting surface of the m.

IV. Discussion

The outcome presented in Section III-A reveals an enhanced coverage in the majority of instances compared to the analyzed alternatives with a shorter runtime. The employment of expensive mesh preprocessing penalizes the duration of the alternative methods significantly. The results exposed in Table II shows that some instances, such as the vase, goblet, and bowl improve the coverage by a significant margin, which is likely caused by the deep internal concavity of these containers. Given that the predominant orientation of the faces points to a region where they will not have a direct visibility of the interior, its visibility is restricted to a set of viewpoints with

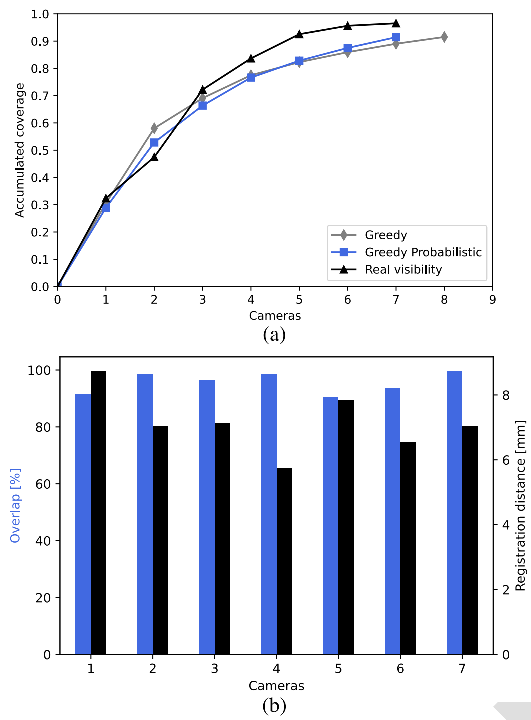


Fig. 13. Evaluated coverage sequence and displacement errors. (a) Accumulated visibility of the Greedy set cover, and the randomized Greedy with the visibility of the scanner. (b) Overlap ratio of the simulated viewpoints and the registered point cloud, including the registration distance in millimeters.

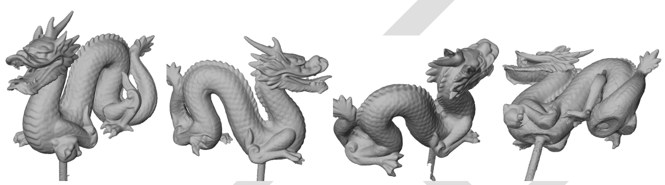


Fig. 14. Reconstructed model rendered from four perspectives based on the seven registered point clouds.

an incidence angle and region of the viewpoint space that is not effectively sampled by these alternative methods. On the contrary, the proposed method samples a subset of cameras that prematurely discards all occluded candidates, ensuring that the subsequent clusterization preserves them by positively weighting their unique visibility. On the other hand, primarily convex objects with reduced curvature, such as the head and bunny, have an increased co-visibility of the surface, resulting in a comparable coverage. Considering the positive results of the proposed method, future instances of the problem could adapt the sampling and clusterization criteria considering other variables which would a priori enable an improved sampling.

The field test has shown that the full system is able to perform with similar results to the simulated problem, even with an average positioning error of 6 mm, yielding an average overlap of 92% of the simulated poses regarding the real captures. The accumulated visibility shown in Fig. 13(a) is higher than the simulated one, which could be associated with multiple factors such as a conservative maximum incident angle and the mutual compensation of the visibility of the whole set of point clouds.

Another aspect to consider is that only one instance of the randomized set cover has been exposed, which has enabled the reduction of one pose with a higher coverage. Future instances of the problem could integrate other objectives in this SCP algorithm factoring the minimum overlap between the captures and the inclusion of other variables to enable the optimization of secondary objectives. The reduced computational cost of the sparse visibility matrix could serve as the basis for the visibility segmentation which could be employed in the positioning of the parts or the design of tooling factoring the visibility. The greedy RTSP employed with the two robotic arms has not been analyzed but it could be extended to systems with multiple independent scanners.

V. Conclusion

In this article, a novel method for the view planning has been introduced based on the efficient sampling of a predefined 3-D model, by employing a sparse representation of the underlying visibility without any expensive mesh preprocessing.

Experiments on a set of 20 complex models have shown that the presented method is nearly 3 times faster than conventional methods, yielding improved coverage with the same number of viewpoints. This method is able to build a sparse representation of the visibility which enables a premature rejection of poor viewpoint candidates. What is more, at the same time prioritizes the sampling of viewpoints covering complex surface patches, without any expensive mesh preprocessing.

Finally, a modified randomized greedy heuristic has been proposed to solve separately the set cover, as well as the sequencing of the robot scanning poses with satisfactory results. This method has been tested with a stereo-structured light scanner mounted on a robot to scan a complex model positioned by another robot. Despite the significant positioning errors accumulated in the kinematic chain, the resulting coverage of the whole set of captures has produced a higher coverage.

Acknowledgment

The authors are grateful to Alberto Izaguirre for its invaluable comments, thorough evaluation, multiple improvements, and extensive analysis, as well as Francisco Javier Huertos for its support and Pedro Álvarez for its assistance.

References

[1] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "OctoMap: An efficient probabilistic 3D mapping framework based on octrees," *Auto. Robots*, vol. 34, no. 3, pp. 189–206, Apr. 2013, doi: 10.1007/s10514-012-9321-0.

[2] A. Bircher, M. Kamel, K. Alexis, H. Oleynikova, and R. Siegwart, "Receding horizon 'next-best-view' planner for 3D exploration," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2016, pp. 1462–1468.

[3] M. S. Ramanagopal, A. P. Nguyen, and J. Le Ny, "A motion planning strategy for the active vision-based mapping of ground-level structures," *IEEE Trans. Autom. Sci. Eng.*, vol. 15, no. 1, pp. 356–368, Jan. 2018.

[4] Y. Li, J. Wang, H. Chen, X. Jiang, and Y. Liu, "Object-aware view planning for autonomous 3-D model reconstruction of buildings using a mobile robot," *IEEE Trans. Instrum. Meas.*, vol. 72, pp. 1–15, 2023.

[5] W. R. Scott, "Model-based view planning," *Mach. Vis. Appl.*, vol. 20, no. 1, pp. 47–69, Jan. 2009, doi: 10.1007/s00138-007-0110-2.

[6] Y.-C. Xu, B. Lei, and E. A. Hendriks, "Camera network coverage improving by particle swarm optimization," *EURASIP J. Image Video Process.*, vol. 2011, pp. 1–10, Jan. 2011.

[7] W. J. Schroeder, J. A. Zarge, and W. E. Lorensen, "Decimation of triangle meshes," *ACM SIGGRAPH Comput. Graph.*, vol. 26, no. 2, pp. 65–70, Jul. 1992.

[8] K. Shimada and D. C. Gossard, "Bubble mesh: Automated triangular meshing of non-manifold geometry by sphere packing," in *Proc. 3rd ACM Symp. Solid Model. Appl.*, Dec. 1995, pp. 409–419.

[9] W. Jing, J. Polden, W. Lin, and K. Shimada, "Sampling-based view planning for 3D visual coverage task with unmanned aerial vehicle," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2016, pp. 1808–1815.

[10] G. H. Tarbox and S. N. Gottschlich, "Planning for complete sensor coverage in inspection," *Comput. Vis. Image Understand.*, vol. 61, no. 1, pp. 84–111, Jan. 1995.

[11] M. Roberts et al., "Submodular trajectory optimization for aerial 3D scanning," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 5334–5343.

[12] V. Chvatal, "A greedy heuristic for the set-covering problem," *Math. Oper. Res.*, vol. 4, no. 3, pp. 233–235, 1979.

[13] C. E. Lemke, H. M. Salkin, and K. Spielberg, "Set covering by single-branch enumeration with linear-programming subproblems," *Oper. Res.*, vol. 19, no. 4, pp. 998–1022, Aug. 1971.

[14] A. Caprara, P. Toth, and M. Fischetti, "Algorithms for the set covering problem," *Ann. Oper. Res.*, vol. 98, nos. 1–4, pp. 353–371, 2000.

[15] S. Sen, "Minimal cost set covering using probabilistic methods," in *Proc. ACM/SIGAPP Symp. Appl. Comput. States Art Pract. (SAC)*, 1993, pp. 157–164.

[16] A. Mavrinac, X. Chen, and J. L. Alarcon-Herrera, "Semiautomatic model-based view planning for active triangulation 3-D inspection systems," *IEEE/ASME Trans. Mechatronics*, vol. 20, no. 2, pp. 799–811, Apr. 2015.

[17] W.-C. Huang, C.-Y. Kao, and J.-T. Horng, "A genetic algorithm approach for set covering problems," in *Proc. 1st IEEE Conf. Evol. Comput., IEEE World Congr. Comput. Intell.*, Jun. 1994, pp. 569–574.

[18] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction To Algorithms*. Cambridge, MA, USA: MIT Press, 2022.

[19] T. A. Feo and M. G. Resende, "Greedy randomized adaptive search procedures," *J. Global Optim.*, vol. 6, no. 2, pp. 109–133, 1995.

[20] G. E. Blelloch, H. V. Simhadri, and K. Tangwongsan, "Parallel and I/O efficient set covering algorithms," in *Proc. 24th Annu. ACM Symp. Parallelism Algorithms Architectures*, Jun. 2012, pp. 82–90.

[21] Q. Wu, W. Zou, and D. Xu, "Viewpoint planning for freeform surface inspection using plane structured light scanners," *Int. J. Autom. Comput.*, vol. 13, no. 1, pp. 42–52, Feb. 2016, doi: 10.1007/s11633-015-0916-8.

[22] J. I. Vasquez-Gomez, L. E. Sucar, and R. Murrieta-Cid, "View/state planning for three-dimensional object reconstruction under uncertainty," *Auto. Robots*, vol. 41, no. 1, pp. 89–109, Jan. 2017.

[23] M. S. Langer and H. H. Bülthoff, "Depth discrimination from shading under diffuse lighting," *Perception*, vol. 29, no. 6, pp. 649–660, Jun. 2000.

[24] E. Gallo, "The SO(3) and SE(3) lie algebras of rigid body rotations and motions and their application to discrete integration, gradient descent optimization, and state estimation," 2022, *arXiv:2205.12572*.

[25] M. Corsini, P. Cignoni, and R. Scopigno, "Efficient and flexible sampling with blue noise properties of triangular meshes," *IEEE Trans. Vis. Comput. Graphics*, vol. 18, no. 6, pp. 914–924, Jun. 2012.

[26] J. L. Bentley, "Multidimensional binary search trees used for associative searching," *Commun. ACM*, vol. 18, no. 9, pp. 509–517, Sep. 1975.

[27] W. D. Maurer and T. G. Lewis, "Hash table methods," *ACM Comput. Surv.*, vol. 7, no. 1, pp. 5–19, Mar. 1975.

[28] J. D. Foley, *Computer Graphics: Principles and Practice*, vol. 12110. Reading, MA, USA: Addison-Wesley, 1996.

[29] J. Bittner, V. Havran, and P. Slavik, "Hierarchical visibility culling with occlusion trees," in *Proc. Comput. Graph. Int.*, Jun. 1998, pp. 207–219.

[30] E. E. Catmull, *A Subdivision Algorithm for Computer Display of Curved Surfaces*. Salt Lake City, UT, USA: The University of Utah, 1974.

[31] E. Lapidous and G. Jiao, "Optimal depth buffer for low-cost graphics hardware," in *Proc. ACM SIGGRAPH/EUROGRAPHICS Workshop Graph. Hardw.*, Jul. 1999, pp. 67–73.

[32] I. Wald, S. Woop, C. Benthin, G. S. Johnson, and M. Ernst, "Embree: A kernel framework for efficient CPU ray tracing," *ACM Trans. Graph.*, vol. 33, no. 4, pp. 1–8, Jul. 2014.

[33] R. Diankov, "Automated construction of robotic manipulation programs," Ph.D. dissertation, Carnegie Mellon Univ., Robot. Inst. Pittsburgh, Pittsburgh, PA, USA, 2010.

[34] J. Pan, S. Chitta, and D. Manocha, "FCL: A general purpose library for collision and proximity queries," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2012, pp. 3859–3866.

[35] N. Mladenović and P. Hansen, "Variable neighborhood search," *Comput. Oper. Res.*, vol. 24, no. 11, pp. 1097–1100, Nov. 1997.

[36] S. Alatartsev, S. Stellmacher, and F. Ortmeier, "Robotic task sequencing problem: A survey," *J. Intell. Robotic Syst.*, vol. 80, no. 2, pp. 279–298, Nov. 2015.

[37] J. Delmerico, S. Isler, R. Sabzevari, and D. Scaramuzza, "A comparison of volumetric information gain metrics for active 3D object reconstruction," *Auto. Robots*, vol. 42, no. 2, pp. 197–208, Feb. 2018.

[38] W. Schroeder, K. Martin, and B. Lorensen, *The Visualization Toolkit*, 4th ed. Kitware, 2006. AQ:7

[39] J. C. K. Chou and M. Kamel, "Finding the position and orientation of a sensor on a robot manipulator using quaternions," *Int. J. Robot. Res.*, vol. 10, no. 3, pp. 240–254, Jun. 1991.

[40] J. J. Kuffner and S. M. La Valle, "RRT-connect: An efficient approach to single-query path planning," in *Proc. IEEE Int. Conf. Robot. Autom.*, vol. 2, Apr. 2000, pp. 995–1001.

[41] I. A. Sucan, M. Moll, and L. E. Kavraki, "The open motion planning library," *IEEE Robot. Autom. Mag.*, vol. 19, no. 4, pp. 72–82, Dec. 2012.

[42] P. J. Besl and N. D. McKay, "Method for registration of 3-D shapes," *Proc. SPIE*, vol. 1611, pp. 586–606, Apr. 1992.

[43] R. B. Rusu and S. Cousins, "3D is here: Point cloud library (PCL)," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2011, pp. 1–4.

[44] S. Rusinkiewicz, "A symmetric objective function for ICP," *ACM Trans. Graph.*, vol. 38, no. 4, pp. 1–7, Aug. 2019.

Benat Urtasun received the master's degree in electrical engineering from Deusto University, Bilbao, Spain, in 2017. He is currently pursuing the Ph.D. degree with the University of Mondragon, Mondragón, Spain.

Since 2018, he has been an Electronic Engineer in robotics and thermography with the LORTEK Technological Center, Basque Research and Technology Alliance (BRTA), Ordizia, Spain. He has specialized in thermography and 3-D inspection for welded metal additive manufacturing and welded components, which is also part of his Ph.D. topic.

Imanol Andonegui received the B.Sc. degree in telecommunication engineering and the M.Sc. and Ph.D. degrees from the University of the Basque Country. His doctoral thesis focused on the characterization, modeling, and linear and nonlinear analysis of nanophotonic materials based on structures with electromagnetic bandgap. AQ:8 AQ:9 AQ:10 AQ:11

He conducted a research stay at the Centre for Ultrahigh Bandwidth Devices for Optical Systems (CUDOS laboratories) belonging to the Australian Research Excellence Network at the University of Sydney. During his research activities, he conducted the first experimental test of optical guidance based on topological theory on silicon nanowires. Later, he joined the Robotics and Computer Vision Research Group, Mondragon University, Mondragón, Spain, where he conducted research related to the dynamic study of robotic manipulators, new sensors for 3-D reconstruction in industrial applications, trajectory planning for part inspection in a thermographic inspection process, and the application of artificial intelligence for predictive maintenance of industrial assets and robotics, as well as the development of AI-based models for optimizing robotic processes.

Eider Gorostegui-Colinas received the Ph.D. degree in applied physics from the University of Navarra (San Sebastian), in 2012, specializing in finite-element method (FEM) modeling of materials during her time at CEIT.

From 2013 to 2015, she actively contributed to the simulation domain at LORTEK, concentrating on FEM simulations to predict distortions and stresses in welding processes. In 2015, she redirected her career toward thermography applied to nondestructive testing (NDT) and process monitoring. Her primary focus has been on advancing inductive thermography. She is presently dedicated to automating this technology and developing algorithms for reliable automatic defect detection. The ultimate goal is to develop a reliable automated technology that can supplant conventional manual methods, leading to improved detection efficiency and enhanced working conditions for technicians.