Curved Surface Inspection by a Climbing Robot: Path Planning Approach for Aircraft Applications

Silya Achat, Julien Marzat and Julien Moras
DTIS, ONERA, Université Paris Saclay, F-91123 Palaiseau, France

Keywords: Coverage Path Planning, Cell Decomposition, Curved Surface Inspection, Sensor Constrained Inspection.

Abstract: This paper presents a path planning method for a climbing robot used for the exterior inspection of an aircraft. The objective is to plan a covering path with the least possible overlap, while respecting constraints related to an embedded sensor, a power cable, and the robot mechanical efforts. To achieve this, a semantic 2D grid model of the aircraft is first created by unfolding and labeling a 3D mesh reference model. An obstacle-based area decomposition method is then applied to divide this 2D discrete space into inspection areas. Inspection segments that satisfy the sensor constraints are then sampled and rearranged based on the order of the areas. The connections between the inspection segments are finally determined by a weighted A^* search approach, so as to limit the gravity-induced mechanical efforts on the robot.

1 INTRODUCTION

Aircraft maintenance is crucial to ensure passenger safety, which includes inspecting their external surfaces to trigger condition-based maintenance operations. Early detection of defects, especially on the fuselage, is necessary to prevent them from causing further damage to the structure. These operations are generally carried out manually by operators, but in recent years the use of remotely operated robotic means has been studied, in particular unmanned aerial vehicles (Bugaj et al., 2020). They allow the fast observation of areas that are difficult to access, while reducing the risks taken by technicians. However, these aerial platforms present some drawbacks. First, they require skilled pilots which limits their development and the use of fleets. Second, they cannot deploy all types of measurement payloads, in particular those requiring proximity such as ultrasound measurements. Finally, they present a risk of damage to the inspected aircraft in case of control loss.

The use of autonomous climbing robots represents an alternative to avoid these issues (Fernandez et al., 2016). Indeed, by moving directly on the surface of the aircraft, these robots will be able to conduct different types of observation (visual, sonar, etc.) while not requiring a qualified pilot. Different locomotion and adherence technologies can be considered such as wheeled robots with ducted fans (Qin et al., 2022; Andrikopoulos et al., 2019) or pneumatic legged configurations (Wang et al., 2019), however magnetic wheels (Starbuck et al., 2021) are not applicable for aircraft surfaces. These climbing robots are generally quite slow, therefore a critical point is to optimize the inspection path allowing to cover the area so that it is the shortest, while taking into account the inspection constraints of the embedded sensors (in range and direction of observation), the motion of the robot (mechanical constraint of adherence or winding of a power cable) and prohibited areas such as the fins.

This paper presents the definition of a method for planning the inspection path for such climbing robots to inspect the fuselage of an aircraft, considering numerous operational and technical constraints. The numerical performances of the procedure are evaluated on a 3D mesh model, as illustrated in Figure 1. This new method relies on a decomposition in areas delimited by obstacles of arbitrary shapes from an unfolded 2D grid model. Inspection segments are defined in these areas with sensor constraints taken into account, and are finally connected using a weighted A^* search procedure to provide the global inspection path.

2 RELATED WORK

Numerous studies have addressed the problem of computing a covering path for applications such as surface inspection (Wu et al., 2015; Liu et al., 2022), as well as in the context of autonomous vacuums

Achat, S., Marzat, J. and Moras, J.
Curved Surface Inspection by a Climbing Robot: Path Planning Approach for Aircraft Applications.
DOI: 10.5220/0012185300003543
In *Proceedings of the 20th International Conference on Informatics in Control, Automation and Robotics (ICINCO 2023) - Volume 1*, pages 433-443
ISBN: 978-989-758-670-5; ISSN: 2184-2809

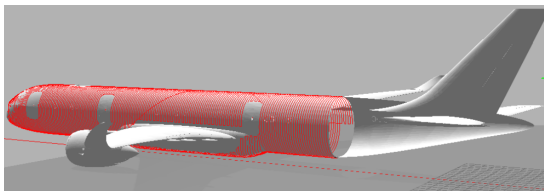


Figure 1: Illustration of a path planned for a climbing robot performing aircraft fuselage inspection.

(Hasan et al., 2014), industrial painting robots (Wu and Tang, 2023; Chen et al., 2008), or problems related to the traveling salesman (Rai et al., 2014; Dahiya and Sangwan, 2018). These paths can be computed on flat or curved surfaces. In the case of curved surfaces, there are two possible approaches: either a path is directly computed in the 3D space based on a 3D model, or a representative 2D parametrization is first extracted from the 3D model to exploit a planar model for path planning.

Various strategies can be applied to obtain a path that fully covers the surface to be inspected. Sampling-based methods generate random paths, leading after many iterations to the complete scanning of a given area (Sörme and Edwards, 2018). On the other hand, deterministic methods can be more time-efficient by dividing the area of interest into sub-areas delimited by obstacles, called cells. This section presents a state of the art of the methods related to the above-mentioned issues, namely path planning methods of 3D structures, different methods for constructing a representative 2D model from a 3D one, and previous works dealing with the decomposition of a zone into sub-areas from obstacles.

Some techniques bypass the 2D representation of space and allow for the construction of covering paths directly on the curved surfaces to be treated. This is the case in the work presented in (Mineo et al., 2017) where parallel paths are calculated on the surface of a 3D mesh model of a curved piece to be followed by a painting robot. The objective is to create paths spaced by a constant distance d from an edge of the piece. Planes Π_i are defined to be perpendicular to the reference edge, containing regularly spaced points P_i. To find the points located at a distance d from the reference edge, we have to simply move from P_i towards the successive intersection points with a mesh of the surface: the curvilinear distance between P_i and an intersection point is the sum of the distances between each intersection point. The point on plane Π_i at a distance d from P_i is determined by linear interpolation between the two successive intersection points having curvilinear distances from P_i respectively greater and lower than d.

Studies on the inspection of large metallic structures such as ship hulls by small differential-drive robots with magnetic wheels (Starbuck et al., 2021) have focused on establishing a planar model representative of a cylinder by stereographic projection. This model has the advantage of being continuous and therefore exploitable for state estimation by a Manifold Invariant Extended Kalman Filter. Indeed, an unrolled model in cylindrical coordinates would present a discontinuity at the angular position $\pm\pi$. However, in the context of planning an inspection path, this kind of cylindrical parametrization would be more appropriate as it involves less deformation of distances between points on the surface. Moreover, this discontinuity of the unrolled model could be exploited to manage directly the non-winding constraint of a robot's power cable during path planning.

Coverage Path Planning (CPP) is a well-known discipline, and a relevant state-of-the-art is presented in (Galceran and Carreras, 2013). A common first step among the methods described is the decomposition of space into cells defined by the obstacles present in the space. The Boustrophedon Decomposition method (Choset and Pignon, 1998) divides a continuous space into cells based on the vertices of the obstacles, which requires the obstacles to be modeled as polygons. This constraint is lifted by the Morse-Based Boustrophedon Cellular Decomposition method (Acar et al., 2002), which defines cells based on critical points on the obstacles depending on the inspection direction. Other methods exploit a discrete environment model represented as a grid, such as the Grid Based Coverage using wavefront algorithm (Zelinsky et al., 1993) and Spiral Spanning-Tree Coverage (Gabriely and Rimon, 2001).

The method described in (Zelinsky et al., 1993) decomposes the space into a grid of uniform cells, where those in contact with an obstacle are not considered by the path planning algorithm. The starting and ending cells are defined, and the algorithm begins the path search from the ending cell. The idea is to propagate a wave from the ending cell to the starting cell, assigning each encountered cell its distance from the ending point. Thus, the cell at the ending point is assigned the index 0. Then, at each iteration, this index is incremented and assigned to all neighboring cells that have not yet been visited. The process is iterated until the starting cell is reached. A covering path is then derived by starting at the starting cell and moving at each iteration to the neighboring cell with the highest assigned number (breaking ties randomly if necessary). Possible applications of this method are adapted to autonomous vacuum cleaners, lawn mowers or security robots.

Another method (Gabriely and Rimon, 2001) is applicable online for a household robot equipped with onboard position, orientation and a range sensor. A decomposition of a large proportion of the free space into so-called mega cells of size $2D$ is performed. Each mega-cell is divided into four sub-cells of the size of the robot D, which are visited only once by the planned path. It is assumed that a square-shaped planar tool is attached to the robot, and is required to cover the entire surface of a workspace. A constraint is that the robot can only move in directions orthogonal to the four sides of this tool which cannot rotate. Starting from an initial cell, the next direction of movement is determined by selecting the unvisited mega cell in its neighborhood (a mega cell is considered visited if at least one of its sub-cells has been visited) in counterclockwise order, resulting in spiraling paths. During this first phase, a spanning tree is built by connecting the mega cells in their order of visit until each mega cell has been visited. Thus, at the end of this recursion, the path will have traversed the outer part of the spanning tree built online. In a second phase, the inner part of the spanning tree is updated by passing through the remaining unvisited sub-cells of the visited mega cells.

However, the use of a sensor during the robotic inspection of an aircraft's surface requires specific observation directions. This excludes the use of the aforementioned Grid Based Coverage using wavefront algorithm (Zelinsky et al., 1993) and Spiral Spanning-Tree Coverage (Gabriely and Rimon, 2001) path planning methods. The Mesh Following Technique (Mineo et al., 2017) method could address this constraint but does not handle obstacle avoidance. An additional constraint comes from the robot's power cable, which should not wrap around the structure. This suggests the use of a 2D unfolded model rather than a 3D one, where the angular discontinuity directly enforces this constraint, unlike a 2D continuous model similar to (Starbuck et al., 2021). Finally, cell decomposition methods such as (Choset and Pignon, 1998) and (Acar et al., 2002) are relevant strategies for optimizing the order of coverage inspection paths. However, these methods would need to be adapted to a discretized environment to be compatible with path planning methods that exploit grids.

3 PROPOSED APPROACH

3.1 Problem Formulation

The problem considered is to generate a path $\mathbb{P}$ as the ordered set of waypoints $\{W_n\}$ that a climbing robot must traverse so that its onboard sensor can scan entirely a given aircraft surface (the focus is put here on the fuselage). The path generation solution must respect constraints related to the sensor embedded on the robot (here, a laser sensor which must scan a locally flat segment is considered), prevent the power cable from tangling, and minimize the gravity-induced efforts on the robot.

By a space transformation, the inspection zone can be represented in 2D and discretised into a semantic grid $\mathcal{G}$ of dimensions $i_{max} \times j_{max}$ cells.

The objective then becomes to find a minimal-length, ordered set of adjacent traversable cells $Q = \{q_1, q_2, ..., q_n\} \subset \mathcal{G}$ such that if the centre of the sensor line projected in $\mathcal{G}$ sequentially traverses each cell $q \in Q$, the sensor will have scanned all the traversable cells. The sensor line is thus considered to be perpendicular to the direction of travel in the grid.

Definition 3.1 (Cell q)**.** Square of a 2D grid with coordinates (i, j) to which a semantic label l is associated. The cell is traversable depending on a user-defined rule on the label defined in Table 1.

Definition 3.2 (Segment)**.** Ordered set of adjacent cells q.

Definition 3.3 (Waypoint W)**.** Each cell $q \in \mathcal{G}$ of the 2D grid corresponds to a waypoint W defined in the 3D space.

First, a method is presented for creating the two-dimensional semantic grid $\mathcal{G}$ corresponding to a simplified and unfolded model of an airplane from a 3D mesh model format, by parametrizing the external surface of the airplane in cylindrical coordinates. Then, a method for planning parallel inspection segments on this unfolded 2D representation is described, such that each segment corresponds to a subsampling of cells q belonging to the grid representing the aircraft surface, and such that the set of these segments $\mathcal{S}$ respects the inspection sensor constraint and ensures complete coverage of the surface. The final set Q is obtained by ordering and connecting the segments using a weighted A^* procedure whose cost takes into account the efforts. The 3D path $\mathbb{P}$ is finally given by associating to each cell $q_n \in Q$ its corresponding waypoint W_n.

3.2 2D Aircraft Model from Unfolded 3D Representation

The outer surface of an aircraft, excluding wings and control surfaces, can be parametrized in cylindrical coordinates with the radius variable representing the surface of interest. Based on this principle, this section proposes a method for creating a 2D semantic

model of an aircraft corresponding to the unfolded model in cylindrical coordinates, which can then be used for path planning of inspection. This discontinuity in the unfolded model implicitly prevents the robot from following a path that would wrap around the aircraft, which would be problematic with respect to the power cable. In this work, planning on (or under) the wings is not addressed, since these flat surfaces do not require any particular preprocessing and path generation is straightforward using a Boustrophedon method (Bähnemann et al., 2021).

Starting from a 3D mesh model of an aircraft, the vertices are extracted. To obtain a point cloud describing only the surface of the aircraft in a more homogeneous manner, the first step is to interpolate points between the vertices. To take into account the semantics of the different parts of the aircraft, each point in the cloud has been manually labeled as indicated in Table 1. A central axis $\vec{z}$ has also been defined, so as to convert the points from Cartesian coordinates to cylindrical coordinates (cf. Figure 2).

The 3D point-cloud is defined in Cartesian frame $\mathcal{P}_{cart} = \{P^i_{cart}\}$ for $i \in \begin{bmatrix}0 & N\end{bmatrix}$ where N is the number of points of the point-cloud. A semantic label l_i is associated to each point of the cloud. We define the cloud $\mathcal{P}_{cyl}$ from $\mathcal{P}_{cart}$ through the cylindrical coordinate transformation:
$\forall P_{cart} = \begin{pmatrix}x & y & z\end{pmatrix}$, $P_{cyl} = \begin{pmatrix}\rho & \theta & z\end{pmatrix}$ where:

$$\rho = \sqrt{x^2+y^2}$$

$$\theta = \begin{cases} \arctan(\frac{y}{x}) & \text{if } x > 0 \\ \arctan(\frac{y}{x}) + \text{sign}(y)\cdot\pi & \text{if } x < 0 \\ \frac{\pi}{2}\cdot\text{sign}(y) & \text{if } x = 0 \text{ and } y \neq 0 \\ \text{indeterminate} & \text{else} \end{cases}$$

An intermediate grid in cylindrical coordinates is defined with steps $\delta\theta$ and δz. It is used to define the two following mappings for each cell:

- $\mathcal{R}(\theta, z)$, which contains a radius ρ as a function of θ and z (see Figure 3a)
- $\mathcal{L}(\theta, z)$, which describes the corresponding label l as a function of θ and z (see Figure 3b)

For a given cell of coordinates $\begin{pmatrix}\theta_f & z_f\end{pmatrix}$ and size $\begin{pmatrix}\delta\theta & \delta z\end{pmatrix}^T$ we consider the set of points $\mathcal{P}_f = \{P_i \in \mathcal{P}_{cyl}\}$ such that $\theta \in \theta_f \pm \frac{\delta\theta}{2}$ and $z \in z_f \pm \frac{\delta z}{2}$. An index i_f is then computed as follows

$$i_f = \arg\max_{P_i \in \mathcal{P}_f} (\rho_i)$$

$$\mathcal{R}(\theta, z) = \rho_{i_f}$$

$$\mathcal{L}(\theta, z) = l_{i_f}$$

If the set $\mathcal{P}_f$ contains several points with identical ρ, the tie is broken using a majority vote on the labels of the set.

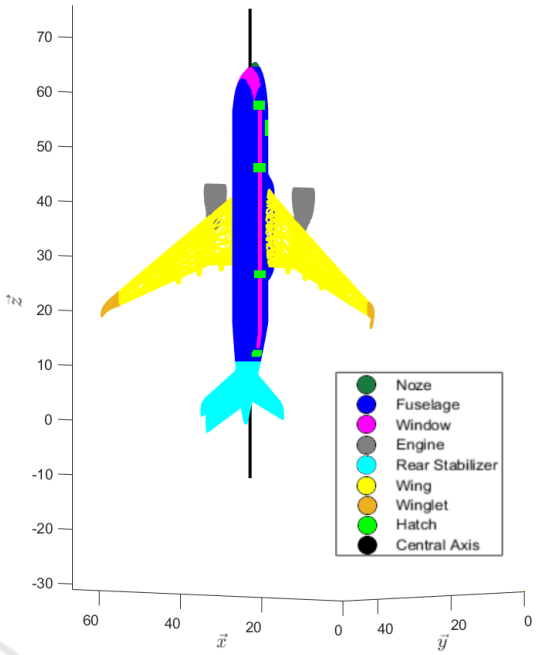


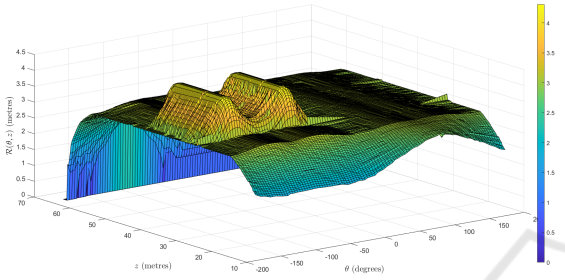
Figure 2: Semantic point cloud obtained after vertex interpolation and manual labeling from an aircraft 3D mesh.

In the aircraft point-cloud considered, points labeled "Winglets", "Engines", and "Rear Stabilizer", that are considered as non-traversable by the robot and which can be easily truncated from the model, were excluded. The "Wing" label points with a radius larger than that of the surrounding fuselage are also excluded, not only because path planning in this area is not addressed as justified previously, but also because their shape is not expected to be compatible with the surface representation based on θ and z. Indeed, several "Wing" labeled points could have highly variable radius, and only the one with the largest radius would be retained in the model, which is not representative of the actual wing surface. Moreover, this would result in the projection of the "Wing" label onto fuselage areas located in the same cell, which would exclude these areas from the inspection path calculation. From the perspective of the fuselage inspection, the resulting "Wing" points are considered as non-traversable obstacles around which the inspection should be conducted.

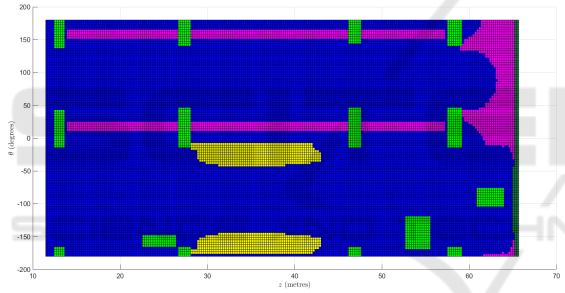
The semantized coordinates of a point $\begin{pmatrix}\rho & \theta & z\end{pmatrix}^T$ and the associated label l in the semantic point cloud can be directly obtained from the grid indices $\begin{pmatrix}i & j\end{pmatrix}^T$ in the unfolded discrete model

Table 1: Semantic labels and related traversability.

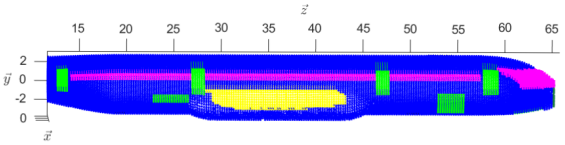
Label l_i	Label Name	Color	is traversable?
l_1	Nose	dark green	0
l_2	Fuselage	blue	1
l_3	Window	magenta	1
l_4	Engine	grey	0
l_5	Rear Stabilizer	cyan	0
l_6	Wing	yellow	0
l_7	Winglet	orange	0
l_8	Hatch	green	0



(a)



(b)



(c)

Figure 3: 3a : Airplane Surfacic Model $\mathcal{R}(\theta,z)$, 3b : Airplane Semantic Unfolded Model $\mathcal{L}(\theta,z)$, 3c : 3D Semantic Point Cloud reconstructed by inverse transformation (1).

through the inverse transformation:

$$\begin{cases} \theta = i \cdot \delta\theta \\ z = j \cdot \delta z \\ \rho = \mathcal{R}(\theta,z) \\ l = \mathcal{L}(\theta,z) \end{cases} \quad (1)$$

In the model studied, the z-axis origin has been repositioned such that the points labeled as "Rear Stabilizer" are excluded from the 2D grid. This inverse transformation makes it possible to reconstruct a 3D point cloud from the two surface models, as illustrated in Figure 3c. This will allow us then to obtain the waypoints in the 3D space from the set of q cells in the 2D grid. Three 2D grid layers of dimension $i_{max} \times j_{max}$ can now be defined, as follows.

Definition 3.4 (Label Grid $\mathcal{G}$). The label at coordinates (i,j) of the semantic grid $\mathcal{G}$ is valued as:

$$\mathcal{G}(i,j) = \mathcal{L}(i \cdot \delta\theta, j \cdot \delta z) \quad (2)$$

Definition 3.5 (Traversability Grid $\mathcal{B}$). $\mathcal{B}$ is a binary grid where

$$\mathcal{B}(i,j) = \begin{cases} 1 & \text{if } \mathcal{G}(i,j) \text{ is traversable} \\ 0 & \text{otherwise} \end{cases} \quad (3)$$

Then, to avoid collisions between the inspection robot and obstacles, a dilation is applied to $\mathcal{B}$ so that the cells with a non-traversable label are dilated by a circular structuring element with radius equal to Δj, which is the length of the sensor field projected onto the grid.

Definition 3.6 (Obstacle Grid O). O is an obstacle grid of dimensions $i_{max} \times j_{max}$ obtained by using the Connected-Component Labeling algorithm (Chang et al., 2004) on the grid $\mathcal{B}$. As a result, neighbor cells $q \in O$ corresponding to obstacles are assigned the same index, and cells corresponding to free space are assigned 0.

3.3 Inspection Segments

From the two-dimensional grid model of the aircraft, a full-coverage path should be determined to provide successive inspection points fulfilling the required objectives and constraints. The climbing robot must perform a full coverage of the surface while avoiding obstacles such as hatches and wings. The robot's sensor field of view should also be aligned with the surface to carry out its mission. This results in the need for the inspection paths to be wrapped around the aircraft's longitudinal axis, each of them remaining in a plane defined by a constant z coordinate. In a 2D grid, this corresponds to scanning the cells in a direction collinear with $\vec{i}$. The definition of the inspection segments is performed thanks to the binary grid $\mathcal{B}$. They are sampled along the i axis between labels considered as obstacles, and they are spaced from each other along the j axis by a distance of $\Delta j = \text{floor}\left(\frac{l_{sensor}}{2 \cdot \delta z}\right)$, where l_{sensor} is the width scanned by the sensor on the surface. This Δj parameter could be chosen smaller than this value in order to pre-compensate for path tracking errors by the robot. Each cell q of a segment (corresponding to coordinates (i,j)) is stored in a vector p_k, which corresponds to a path identified by index

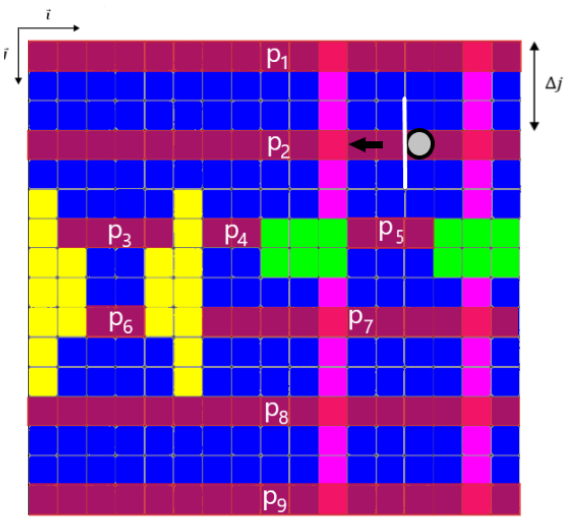


Figure 4: Definition of the inspection segments p_k visualized on the $\mathcal{G}(i,j)$ grid (simplified version without dilatation of the obstacles). The robot is represented in grey, and the field of its inspection sensor is shown in white. The direction of the robot's movement is represented by a black arrow.

k (see Figure 4), and the collection of these segments is stored in a set of inspection segments $\mathcal{S}$.

Definition 3.7 (Inspection Segment p_k)**.** A segment p_k of index $k \in \mathbb{N}^*$, and of first cell q_b and last cell q_e with respective coordinates (i_b, j_b) and (i_e, j_e), verifies:

- $\forall q_1, q_2 \in p_k\ ,\ j_1 = j_2 := j_k$
- if $i_b > 1$, then $\mathcal{B}(i_b - 1, j_k) = 0$
- if $i_e < i_{max}$, then $\mathcal{B}(i_e + 1, j_k) = 0$

The collection of these segments is initially stored in a set of inspection segments $\mathcal{S}$.

3.4 Obstacle-Based Area Decomposition in 2D Discrete Space

The grid $\mathcal{G}$ is partitioned into ordered areas a_m with respect to obstacles from the grid O. This will be used to connect the inspection segments contained in these areas.

Definition 3.8 (Area a_m)**.** The m-th area a_m is a set of traversable cells located between groups of non-traversable cells with the same obstacle index, respectively denoted by n_{ol} on the left border and n_{or} on the right border. The areas are primarily ordered along ascending i indices, and then along the j indices. If there exists an area a_i, $i < m$ that has the same n_{ol} and n_{or} indices as a_m, then a third index cpt is used to distinguish them. This defines the unique identifier vector $\alpha_m = [n_{ol}\, n_{or}\, cpt]$ of the area a_m. The ordered set $\mathcal{A}$ stores all the identifiers α_m.

The obstacle indices n_{ol} and n_{or} take their values from the grid O, and if there is no obstacle (i.e. a traversable cell is located at the minimum or maximum i index on a given j-th line) then the default assigned index is -1.

Remark. The variable $cpt \in \mathbb{N}$ is a counter to distinguish areas that are located between the same obstacles of index n_{ol} and n_{or} but are not connected along the j-axis.

All the segments p_k that belong to an area a_m are traversed in ascending order of j to determine the ordered set of cells Q_m. It should be noted that this assumes that the very starting cell of the global path will be in position $(i = 1, j = 1)$. Let q_b and q_e be the starting and ending cells of p_k. To determine the area a_m to which a path p_k belongs, we query the labeled obstacle grid O to find the obstacle identifiers n_{ol} and n_{or} respectively neighbors of q_b and q_e. Indeed, we keep track of the last j-coordinate of the last added path to each area a_m, and this data is stored in a vector $last\_j\_area$ at the index m. Let q_{last} be the last cell q of the last segment p_k added in Q_m. If q_{last} is closer in the grid to q_e than q_b, the next segment $p_k = \{q_1, q_2, ..., q_N\}$ is reversed. This operation consists of reordering p_k in descending order of indices n, resulting in a segment $p_k = \{q_N, q_{N-1}, ..., q_1\}$. This procedure iterates until all the segments $p_k \in \mathcal{S}$ have been inserted into Q. As a result, on average, one inspection segment over two is reversed to sweep the surface in a Boustrophedon pattern.

Algorithm 1 describes the corresponding pseudo-code, and an illustration of the clustering is given in Figure 5. The remaining step of the method is to connect the inspection segments together to form the final global path Q.

3.5 Path Connection Between Inspection Segments that Minimize Gravity Constraints

The robot's configuration implies that it experiences the least amount of effort when its vertical axis is parallel to gravity, that is to say when it adheres to a horizontal surface. Indeed, the robot tends to slip the most when it adheres to a vertical surface, which requires it to deliver more power to adhere more strongly to the wall and thus increase the frictional force that keeps it in place. It will therefore expend less energy adhering to the surface at the top ($\theta = \frac{\pi}{2}$) and bottom of the aircraft ($\theta = -\frac{\pi}{2}$). Moreover, intermediate cells need to be inserted outside of the inspection segments so that

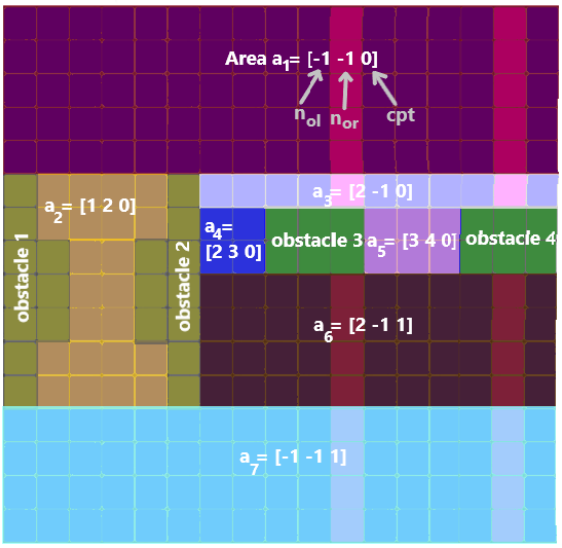


Figure 5: Example of areas numbering based on their semantics and surrounding obstacles. When there are no obstacles to the left of an area, n_{ol} defaults to -1 (and vice versa for n_{or}). In this example, we can observe that a_1 and a_2 are differentiated by their value of cpt since the connection criterion is not satisfied, same for a_3 and a_6.

the robot can move from an inspected area to an uninspected one, and it can also move from one segment to another within the same area. It is preferable for the robot to pass through the locations where the gravitational force is energetically less constraining. For this purpose, a weighted version of the A^* algorithm is used to plan the connecting segment (defined below) linking the last cell of the previous inspected segment q_{last} and the first cell q_b of a not-yet-inspected segment.

Definition 3.9 (Connecting Segment d_{k_1,k_2}). A Connecting Segment d_{k_1,k_2} of indices $k_1, k_2 \in \mathbb{N}^*$ is a segment that connects two inspection segments p_{k_1} and p_{k_2}. Let q_{b_d} and q_{e_d} be the first and last cell of d_{k_1,k_2}, and q_{b_k} and q_{e_k} be the first and last cell of p_k. Then d_{k_1,k_2} verifies $q_{b_d} = q_{e_{k_1}}$ and $q_{e_d} = q_{b_{k_2}}$.

The computation process evaluates the cost f associated to the starting cell and other neighboring cells successively until reaching the q_b cell (the neighborhood is defined as 8-connectivity). These costs between a currently evaluated node q_{cur} and one of its neighbors q_{nei} are defined as:

$$g(q_{nei}, \mathcal{L}) = g(q_{cur}, \mathcal{L}) + c(q_{nei}, \mathcal{L}) \cdot \|\xi_{nei} - \xi_{cur}\| \tag{4}$$

$$f(q_{nei}, q_b, \mathcal{L}) = g(q_{nei}, \mathcal{L}) + \|\xi_b - \xi_{cur}\| \tag{5}$$

where ξ_{cur}, ξ_{nei}, ξ_b are the discrete indices (i, j) associated with cells q_{cur}, q_{nei}, q_b, respectively. The

Algorithm 1: Ordering of the inspection segments.

Input: set $\mathcal{S}$, grid O
Output: set Q

sets $\mathcal{A} \leftarrow \{\}$, $Q \leftarrow \{[]\}$
Vector $last\_j\_area \leftarrow []$
for p_k *in* $\mathcal{S}$ **do**
  Integer $cpt \leftarrow 0$
  cells $q_b \leftarrow p_k.begin()$, $q_e \leftarrow p_k.end()$
  if $q_b.i > 1$ **then**
    $n_{ol} \leftarrow O(q_b.i - 1, q_b.j)$
  else
    $n_{ol} \leftarrow -1$
  if $q_e.i < i_{max}$ **then**
    $n_{or} \leftarrow O(q_e.i + 1, q_e.j)$
  else
    $n_{or} \leftarrow -1$
  Vector $\alpha \leftarrow [n_{ol} \quad n_{or} \quad cpt]$
  Integer $m \leftarrow \mathcal{A}.find(\alpha)$
  if *m is empty* **then**
    $\mathcal{A}.pushBack(\alpha)$
    $m \leftarrow \mathcal{A}.find(\alpha)$
  else
    while $|q_b.j - last\_j\_area[m]| \neq \Delta j$ *and* *m not empty* **do**
      $cpt \leftarrow cpt + 1$
      $\alpha \leftarrow [n_{ol} \quad n_{or} \quad cpt]$
      $m \leftarrow \mathcal{A}.find(\alpha)$
  $q_{last} = Q\{m\}.end().end()$
  if $distance(q_{last}, q_e) < distance(q_{last}, q_b)$ **then**
    p_k.reverse();
  $Q\{m\}.pushBack(p_k)$
  $last\_j\_area[m] \leftarrow q_b.j$

weighting function is defined as :

$$w(q_{nei}, \mathcal{L}) = \begin{cases} 1 + |cos(\theta)| & \text{if } \mathcal{L}(\theta, z) \text{ is traversable} \\ \infty & \text{otherwise} \end{cases} \tag{6}$$

where (θ, z) are the non-discretized coordinates associated to q_{nei}. θ is defined such that $\theta \in [-\pi; \pi]$ and such that $\theta = -\frac{\pi}{2}$ at the bottom of the aircraft, and $\theta = \frac{\pi}{2}$ at the top. This cost factor gives more weight to the distance between a node q_{cur} and its neighbor q_{nei} if it has an angular position θ far from $\pm\frac{\pi}{2}$. It would also be possible to introduce a cost function that prioritizes a path crossing through certain labels over others, for example, in cases where avoiding passage over windows is preferable, without being an absolute requirement. This is a way to generate a semantic-aware path from the available graph information, as previously proposed in (Achat et al., 2022).

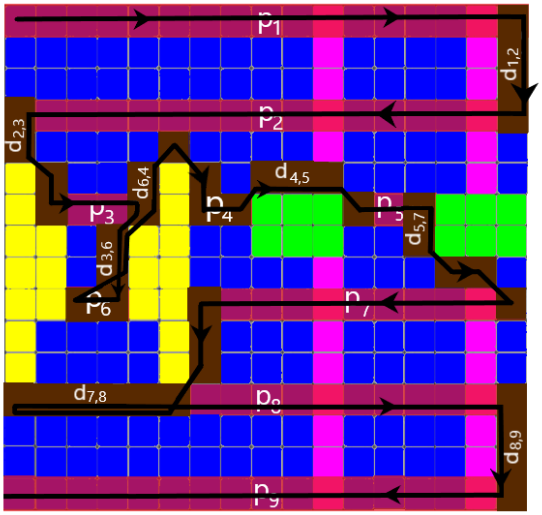


Figure 6: Final global path Q(black) vizualized in the 2D unfolded discrete space (simplified version). It is composed of inspection segments p_k (red) and connecting segments d_{k_1,k_2} (brown). In this example, $Q = \{Q_1, Q_2, \ldots, Q_7\}$, with $Q_1 = \{p_1, d_{1,2}, p_2, d_{2,3}\}, Q_2 = \{p_3, d_{3,6}, p_6, d_{6,4}\}, Q_3 = \{\}, Q_4 = \{p_4, d_{4,5}\}, Q_5 = \{p_5, d_{5,7}\}, Q_6 = \{p_7, d_{7,8}\}, Q_7 = \{p_8, d_{8,9}, p_9\}$.

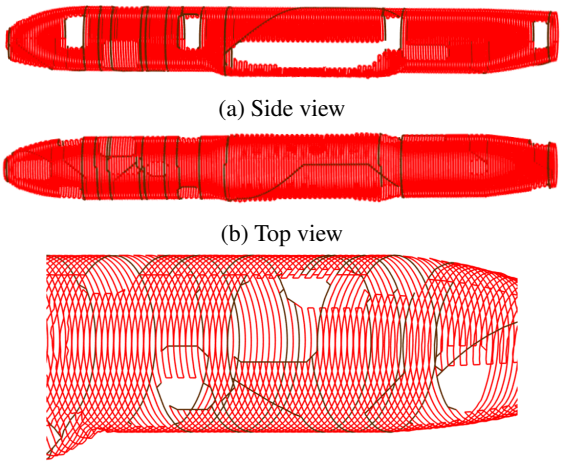
All the inspection segments previously ordered by areas are traversed in this order to be connected together in Q. Thus, the Global Path Q contains the subsets Q_m ranked in ascending order of m such as $\forall Q_m \in Q$ and $\forall p_k \in Q_m$, if p_k belongs to a_m. The p_k are ranked in Q_m in ascending order of k. Every subsets $Q_m \in Q$ contains also connecting segments d_{k_1,k_2} between every p_k such as $\forall d_{k_1,k_2} \in Q_m : \exists! p_k \in Q_m | q_{b_d} = q_{e_k}$, with q_{b_d} the first cell of d_{k_1,k_2} and q_{e_k} the last cell of p_k. d_{k_1,k_2} is inserted at the index following p_k in Q_m.

In the end, a comprehensive global path which includes all the inspection segments is obtained, with additional connecting paths (see Figure 6). The spacing between these segments is at most equal to half of the sensor's field of view (by construction of the 2D grid), ensuring that the entire space to be inspected is covered. The inverse transformation (1) enables to project back the global inspection path in the initial 3D space as illustrated in Figure 7.

4 NUMERICAL EXPERIMENTS

4.1 Path Planning Unitary Tests

This section aims to evaluate the previously presented path planning method, specifically focusing on the effectiveness of sorting inspection paths into areas



(a) Side view

(b) Top view

(c) Zoomed front bottom view

Figure 7: Global planned path with full coverage. Inspection segments are represented in red, and connecting segments that limit gravity constraints in brown.

Figure 8: Computed inspection path in the case "Without sorting by areas". In this case, the length of connecting paths (represented in brown) is significantly increased in comparison to the path illustrated in Figure 7.

prior to the connection process. The developed algorithms have been implemented in C++ and run within a Ubuntu 20.04 WSL Environment on a computer equipped with Intel Core i5-10400H CPU and 16GB RAM. Calculation times are detailed in Table 2.

Table 2: Calculation times.

Sampling of inspection segments p_k (ms)	0.54
Algorithm 1 (ms)	0.61
Calculation and insertion of connecting segments (ms)	53

The following Table 3 presents the numerical parameters associated to the 2D grid generated from the aircraft 3D mesh. Table 4 presents the characteristics of the inspection segments that have been defined in the 2D grid. It should be noted that in our case, the length of the sensor's field of view is such that its half-length in the 2D grid Δj is equal to 1. This implies that the robot must pass through all the grid cells q associated with traversable labels.

A comparison was conducted between the computed inspection paths in the two following cases:

- "With sorting by area": the global path is computed with the proposed path planning method.
- "Without sorting by area": the global path is com-

Table 3: Numerical characteristics of the test grid.

Longitudinal resolution δz (metres)	0.24
Angular resolution $\delta\theta$ (degrees)	3.6
Grid dimensions $i_{max} \times j_{max}$	101×226
Number of areas	33

puted by connecting the inspection segments together, still avoiding obstacles, but without prior area sorting (Section 3.4).

In both cases, the length of the computed path, as well as the number of connected waypoints were recorded (see Table 5). It turns out that sorting the inspection segments by the areas defined between obstacles allows us to halve the total distance traveled, and decrease by 93% the number of interpolated waypoints to connect these segments.

Table 4: Numerical characteristics of the inspection segments generated from the test grid.

Number of inspection segments sampled	300
Inspection segments cumulative length in pixels (in metres)	19282 (3639 m)
Average spacing between two consecutive waypoints (metres)	0.19
Half-length of the sensor's range Δj in cells $\left(\frac{l_{sensor}}{2} \text{ in metres}\right)$	1 cell (0.4 m)

Table 5: Comparison of interpolated paths with and without sorting inspection paths into areas.

Method	Without sorting by areas	With sorting by areas
Number of interpolated waypoints	19234	1343
Path length (in metres)	8206	4034

4.2 ROS Simulation

For illustration purposes and to eventually close the gap with real-world deployment, a ROS simulation has been set up to test the path-following behaviour by a climbing robot on the surface of an aircraft. A wheel-based robot equipped with an adhesion system similar to the Vortex Climbing Robot (Andrikopoulos et al., 2019) is simulated using the ROS/Gazebo environment (see Figure 10)[1]. The adhesive force of the robot is emulated by applying forces on the robot in the opposite direction of the local surface normal.

[1]Video: https://tinyurl.com/OneraClimbingRobotPlanning

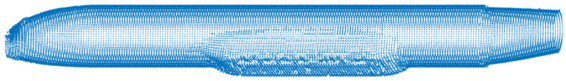


Figure 9: 3D field of surface normal vectors to the airplane obtained through the inverse transformation of the grid $\mathcal{N}$.

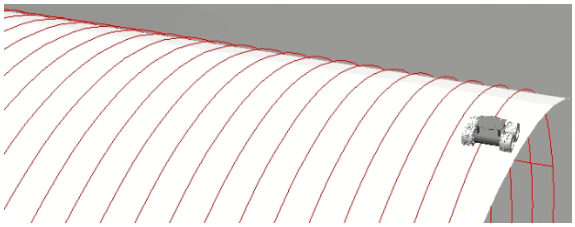


Figure 10: Climbing wheeled robot tracking the generated inspection path on the surface of an aircraft, simulated in a ROS/Gazebo environment.

For this purpose, a model of the surface normal vectors of the airplane $\mathcal{N}(i,j)$ has been established from a meshed model, in a similar manner to the models $\mathcal{R}(i,j)$ and $\mathcal{L}(i,j)$. Note that $\mathcal{N}$ is a grid of dimensions $i_{max} \times j_{max} \times 3$. A representation of the surface normal vectors of the airplane is depicted in Figure 9.

In order to perform path tracking control, the 3D position and orientation of the robot $(x_r \quad y_r \quad z_r \quad quat)^T$ are associated with a 2D position and orientation in the plane defined by (θ, z). The orientation of the robot in this plane corresponds to the projected yaw angle ψ, and its position is defined by θ and z. The odometry data provides the position of the robot in the 3D space, namely x_r, y_r and z_r, as well as its orientation *quat* parametrized as a quaternion. Thus, the robot's state in the 2D plane is described by the coordinates $(\theta \quad z \quad \psi)^T$.

This robot local planar state is determined as follows. Firstly, the unit vector $\vec{v}$ corresponding to the robot's orientation in the 3D space is computed using the *quat* value. Let $P_{1,cart}$ be the origin of the vector $\vec{v}$ (thus $P_{1,cart}$ is located at the center of the robot in the 3D space) and let $P_{2,cart}$ be the point located at its endpoint. These two points can be rewritten in cylindrical coordinates similarly to Section 3.2, which gives $P_{1,cyl}$ and $P_{2,cyl}$. The coordinates of these points in the local 2D plane are $P_1 = (\rho_1, \theta_1, z_1)$ and $P_2 = (\rho_2, \theta_2, z_2)$, which allows us to determine the state of the robot as:

$$\begin{pmatrix} \theta \\ z \\ \psi \end{pmatrix} = \begin{pmatrix} \theta_1 \\ z_1 \\ \text{atan2}(z_2 - z_1, \rho_2 \cdot \theta_2 - \rho_1 \cdot \theta_1) \end{pmatrix} \quad (7)$$

Due to the discontinuity at $\theta = \pm\pi$, the calculated value of the angle ψ can be incorrect when the points $P_{1,cyl}$ and $P_{2,cyl}$ are located on different sides of this boundary. To address this issue, the value of θ_2 is adjusted within the range $[-2\pi; +2\pi]$ based on

$\theta_1 \in [-\pi;+\pi]$. The cells $q = (i, j)$ of the precomputed offline global inspection path Q are converted into waypoints with coordinates

$$\begin{pmatrix}\theta_w & z_w & \rho_w\end{pmatrix} = \begin{pmatrix}i \cdot \delta\theta & j \cdot \delta z & \mathcal{R}(i \cdot \delta\theta, j \cdot \delta z)\end{pmatrix}$$

The path tracking performed by the robot is a waypoint-based navigation method based on a unicycle model. Thus, the robot aligns itself towards the next waypoint by adjusting its yaw angle in the 2D plane with the angular set-point $\psi_s = \text{atan2}(z_w - z, \rho_w \cdot \theta_w - \rho.\theta)$ using a proportional angular controller. A velocity controller is then applied once the angular error is below a small threshold. This control logic leads to the expected correction in the 3D space and the simulated robot model was successfully able to track the inspection path on the surface of the aircraft (Figure 10).

5 CONCLUSIONS AND PERSPECTIVES

A method for planning a covering path on a curved surface in the context of an automated inspection mission has been proposed in this paper. A 2D unfolded model of this surface was established using a parametrization of an input 3D model in cylindrical coordinates. It was discretized to obtain two surface functions, one describing the shape of the surface and the other its semantics. Inspection segments were sampled in this 2D space, taking into account various constraints related to the robot. A method for decomposing this discrete space into areas has been defined to order these segments efficiently. Connection segments have been inserted between the inspection segments and calculated to pass through locations that minimize the robot's efforts using a weighted $A*$ search procedure. This method has been tested numerically on a representative aircraft 3D mesh where the relevance of sorting the inspection path segments by areas was demonstrated, and in a Gazebo simulation with a simplified climbing robot model.

In the context of aircraft inspection, approximating its surface with a cylindrical parametrization presents some limitations, e.g. to exclude the wings in the model. It would be interesting to investigate in future work the extension of the path planning method to surfaces of different shapes, or the possibility of dividing a 3D model into multiple sub-models that can be individually approximated by a collection of geometrical primitives.

ACKNOWLEDGMENTS

This research was partially supported by DGAC France Relance project EXAM, in the frame of the NextGenerationEU program.

REFERENCES

Acar, E. U., Choset, H., Rizzi, A. A., Atkar, P. N., and Hull, D. (2002). Morse decompositions for coverage tasks. *The International Journal of Robotics Research*, 21(4):331–344.

Achat, S., Marzat, J., and Moras, J. (2022). Path planning incorporating semantic information for autonomous robot navigation. In *19th International Conference on Informatics in Control, Automation and Robotics (ICINCO)*, Lisbonne, Portugal.

Andrikopoulos, G., Papadimitriou, A., Brusell, A., and Nikolakopoulos, G. (2019). On model-based adhesion control of a vortex climbing robot. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1460–1465.

Bähnemann, R., Lawrance, N., Chung, J. J., Pantic, M., Siegwart, R., and Nieto, J. (2021). Revisiting boustrophedon coverage path planning as a generalized traveling salesman problem. In *12th International Conference on Field and Service Robotics*, pages 277–290.

Bugaj, M., Novák, A., Stelmach, A., and Lusiak, T. (2020). Unmanned aerial vehicles and their use for aircraft inspection. In *2020 New Trends in Civil Aviation (NTCA)*, pages 45–50.

Chang, F., Chen, C.-J., and Lu, C.-J. (2004). A linear-time component-labeling algorithm using contour tracing technique. *Computer Vision and Image Understanding*, 93:206–220.

Chen, H., Fuhlbrigge, T., and Li, X. (2008). Automated industrial robot path planning for spray painting process: A review. In *IEEE International Conference on Automation Science and Engineering*, pages 522–527.

Choset, H. and Pignon, P. (1998). Coverage path planning: The boustrophedon cellular decomposition. In *Field and service robotics*, pages 203–209.

Dahiya, C. and Sangwan, S. (2018). Literature review on travelling salesman problem. *International Journal of Research*, 5:1152–1155.

Fernandez, R. F., Keller, K., and Robins, J. (2016). Design of a system for aircraft fuselage inspection. In *IEEE Systems and Information Engineering Design Symposium*, pages 283–288.

Gabriely, Y. and Rimon, E. (2001). Spanning-tree based coverage of continuous areas by a mobile robot. *Annals of mathematics and artificial intelligence*, 31:77–98.

Galceran, E. and Carreras, M. (2013). A survey on coverage path planning for robotics. *Robotics and Autonomous Systems*, 61:1258–1276.

Hasan, K. M., Reza, K. J., et al. (2014). Path planning algorithm development for autonomous vacuum cleaner robots. In *International Conference on Informatics, Electronics & Vision (ICIEV)*.

Liu, Y., Zhao, W., Liu, H., Wang, Y., and Yue, X. (2022). Coverage path planning for robotic quality inspection with control on measurement uncertainty. *IEEE/ASME Transactions on Mechatronics*, 27:3482–3493.

Mineo, C., Pierce, S., Nicholson, I., and Cooper, I. (2017). Introducing a novel mesh following technique for approximation-free robotic tool path trajectories. *Journal of Computational Design and Engineering*, 4:192–202.

Qin, L., Jin, Z., Suo, S., Zhang, C., Qiao, K., and Liu, J. (2022). Design and research of a wall climbing robot with ducted fan for aircraft appearance inspection. In *International Conference on Sensing, Measurement & Data Analytics in the era of Artificial Intelligence (ICSMD)*.

Rai, K., Madan, L., and Anand, K. (2014). Research paper on travelling salesman problem and its solution using genetic algorithm. *International Journal of Innovative Research in Technology*, 1(11):103–114.

Sörme, J. and Edwards, T. (2018). A comparison of path planning algorithms for robotic vacuum cleaners. In *BSc report, KTH Royal Institute of Technology*.

Starbuck, B., Fornasier, A., Weiss, S., and Pradalier, C. (2021). Consistent state estimation on manifolds for autonomous metal structure inspection. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 10250–10256.

Wang, C., Gu, J., and Li, Z. (2019). Switching motion control of the climbing robot for aircraft skin inspection. In *IEEE International Conference on Fuzzy Systems*.

Wu, H. and Tang, Q. (2023). Robotic spray painting path planning for complex surface: boundary fitting approach. *Robotica*, 41(6):1794–1811.

Wu, Q., Lu, J., Zou, W., and Xu, D. (2015). Path planning for surface inspection on a robot-based scanning system. In *IEEE international conference on mechatronics and automation (ICMA)*, pages 2284–2289.

Zelinsky, A., Jarvis, R. A., Byrne, J. C., and Yuta, S. (1993). Planning paths of complete coverage of an unstructured environment by a mobile robot. In *Proceedings of International Conference on Advanced Robotics*, volume 13, pages 533–538.