

Article

# Smooth Complete Coverage Trajectory Planning Algorithm for a Nonholonomic Robot

Ana Šelek , Marija Seder \* , Mišel Brezak and Ivan Petrović 

Laboratory for Autonomous Systems and Mobile Robotics (LAMOR), Faculty of Electrical Engineering and Computing, University of Zagreb, 10000 Zagreb, Croatia

\* Correspondence: marija.seder@fer.hr

**Abstract:** The complete coverage path planning is a process of finding a path which ensures that a mobile robot completely covers the entire environment while following the planned path. In this paper, we propose a complete coverage path planning algorithm that generates smooth complete coverage paths based on clothoids that allow a nonholonomic mobile robot to move in optimal time while following the path. This algorithm greatly reduces coverage time, the path length, and overlap area, and increases the coverage rate compared to the state-of-the-art complete coverage algorithms, which is verified by simulation. Furthermore, the proposed algorithm is suitable for real-time operation due to its computational simplicity and allows path replanning in case the robot encounters unknown obstacles. The efficiency of the proposed algorithm is validated by experimental results on the Pioneer 3DX mobile robot.

**Keywords:** mobile robot; complete coverage; path planning; path smoothing; velocity profile optimization



**Citation:** Šelek, A.; Seder, M.; Brezak, M.; Petrović, I. Smooth Complete Coverage Trajectory Planning Algorithm for a Nonholonomic Robot. *Sensors* **2022**, *22*, 9269. <https://doi.org/10.3390/s22239269>

Academic Editor: Andrey V. Savkin

Received: 19 October 2022

Accepted: 23 November 2022

Published: 28 November 2022

**Publisher's Note:** MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



**Copyright:** © 2022 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

## 1. Introduction

The task of a Complete Coverage Path Planning (CCPP) algorithm is to generate such a path for a mobile robot that ensures that the robot completely covers the entire environment while following the planned path. There are many real-world applications that require a CCPP algorithm, such as floor cleaning [1–3], demining [4,5], automated harvesting [6], lawn mowing [7], autonomous underwater exploration [8,9], etc. To achieve efficient coverage in these applications, the planned path must satisfy some requirements, where the most important ones are maximizing the coverage rate, minimizing the complete coverage time, path length, overlap area, and energy consumption of the robot, and rapid path replanning if the environment changes during the execution of the coverage task. Many CCPP algorithms have been developed, but none of them meet all the above requirements, so there is still room for improvement.

In this paper, we propose a CCPP algorithm which generates smooth complete coverage paths that allow nonholonomic mobile robots to move in optimal time while following them (we called it SCCPP). Moreover, the proposed SCCPP algorithm is suitable for real-time operation due to its computational simplicity and allows path replanning in case the robot encounters unknown obstacles. The existing coverage algorithms in the literature are either non-smooth so they have increased coverage redundancy due to the non-ideal path following, or they have slow path planning and replanning. The SCCPP algorithm combines two of our previous works: the fast coverage planning algorithm [10] with the fast clothoid calculation [11]. The first algorithm is our replanning spanning tree coverage (RSTC) algorithm that generates a path in a low-resolution occupancy grid map to reduce the computational complexity and minimize the overlap rate. The path is the shortest possible coverage path in the corresponding graph, which contains sharp 90° turns. To avoid stopping and rotating the robot at turning points, we used clothoids to smooth the path generated by the RSTC algorithm. The main advantage of the clothoids is the linear

change in their curvature, which allows the nonholonomic robot to move in a time-optimal manner while following such a smoothed path.

The SCCPP is the real-time traversable collision-free complete coverage path planning algorithm based on clothoids, which gives minimal path length, the coverage time, and overlap area and maximal coverage rate compared to the state-of-the-art coverage algorithms. Such a path is suitable and feasible for nonholonomic mobile robots.

The remaining of the paper is structured as follows. Section 2 presents the overview of the complete coverage path planning algorithms. The proposed smooth complete coverage path planning algorithm is described in Section 3. The simulation and experimental results are given in Sections 4 and 5, respectively, followed by the conclusion of the paper.

## 2. Related Work

One of the earliest works on complete coverage path planning is presented in [12], which defines requirements for covering a continuous two-dimensional (2D) environment. Typically, methods for CCPP tasks assume that the environment is known, including the obstacle configuration. A number of methods perform an exact cell decomposition of the environment, e.g., [13–15], where free space is decomposed into trapezoidal, rectangular, or triangular cells. The path that completely covers each cell (regarding their shape) can lead to increased coverage redundancy and is time-consuming when the environment changes. Simpler and faster are methods that perform an approximate cell decomposition where the environment is represented with a 2D occupancy grid map, i.e., equally distributed square grid of discrete cells [16–19].

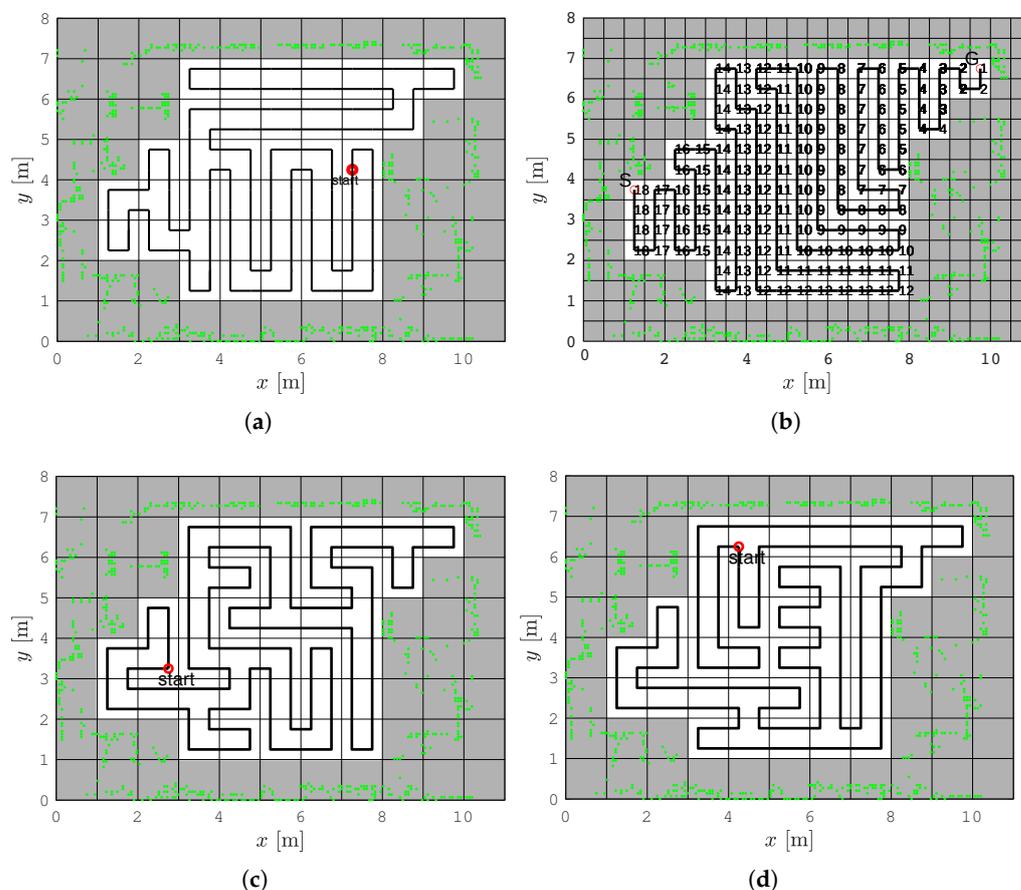
The size of the square grid cells directly affects the replanning rate and the coverage rate. These two requirements are opposite, i.e., larger cell sizes allow real-time replanning due to lower computational complexity, while smaller ones ensure higher coverage rate at the cost of higher computational complexity. Fast replanning algorithms usually select the cell size equal to the footprint of the robot, while the cell size is much smaller in the algorithms that ensure a high coverage rate, usually from 2 to 10 cm for the cell side [20,21].

A graph can be constructed from the occupancy grid map, where the grid cells are the nodes and the connections between adjacent grid cells are the edges. An optimal CCPP would ensure that the robot completely covers the entire environment by visiting all nodes in the graph only once, but this is a NP-hard problem, known as the Traveling Salesman Problem (TSP) [22]. Therefore, CCPP algorithms use approximate or heuristic solutions (see Figure 1). The complete coverage path planning algorithm based on a numeric potential field approach called the path transform (PT) can be used for the TSP solution [16]. Ant colony optimization (ACO) [23,24] is based on artificial ants colony that are able to generate the shortest feasible path of the TSP successively. Another solution for TSP can be determined by using the Hopfield neural network (HNN) for an optimization that consists of a single layer containing one or more fully connected recurrent neurons [25–27]. The Spanning Tree Coverage (STC) algorithm [28] and the Replanning Spanning Tree Coverage (RSTC) algorithm [10] produce the optimal coverage path in linear time. The CCPP algorithms on a high-resolution occupancy grid map have increased coverage rate.

The Complete Coverage  $D^*$  (CCD $^*$ ) algorithm [2] has a high coverage rate, which leads to increased coverage redundancy when the environment changes. The hex-decomposition-based coverage planning (HDCP) algorithm [29] for unknown environments plans smooth paths for Dubins vehicles to follow at constant velocity in real-time. Adaptive coverage path planning [30] is aiming to achieve complete coverage with minimal path length and it is efficient in dynamic environments. Complete coverage path planning based on biologically inspired neural network [31] plans collision-free trajectory for real-time coverage task in dynamic environments. The energy constrained online coverage path planning [32] is based on contour following, which causes sharp turns in rectangular environments. The drawbacks of the above algorithms are that the coverage redundancy is increased when dynamic obstacles are present, and a trade-off exists between a higher coverage rate and higher coverage redundancy. Moreover, the energy consumption and the coverage

time are high because most of these algorithms generate the path with straight lines that form sharp turns. When the nonholonomic robot follows such a path, it must stop and reorient itself so that the heading of the robot matches the path direction [33]. Authors in [34] do a complete 2D sweep coverage, but their approach does not produce smooth paths, is not computationally efficient, and does not consider dynamic obstacles but only static environments.

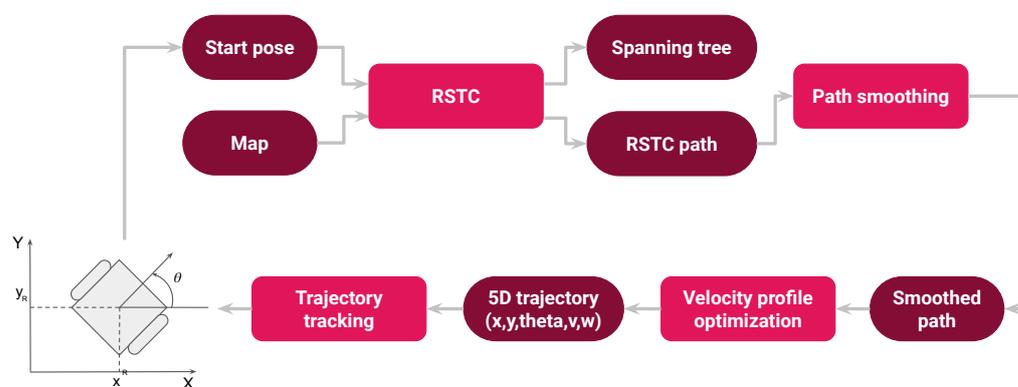
To provide optimal and feasible paths with curvature continuity that are easy to follow by nonholonomic mobile robots, path smoothing algorithms are used. A smoothing algorithm provides motion continuity and reduces the execution time of coverage tasks. The first research on finding the shortest curvature constrained smooth paths consisting of straight lines and arcs was done by Dubin in [35]. The presence of discontinuities at the intersection of the straight lines and circular arcs makes them infeasible for real applications. Backman et al. [36] presented a curvature continuous CCPP algorithm using the Dubin curve for agricultural vehicles. The generated path is easy to follow for real-world applications and the algorithm is efficient in coverage time but computationally complex. Yu et al. [37] proposed Dubin's vehicle-based coverage algorithm, which minimizes the non-working distance in CCPP, reduces the number of turns of the vehicle, maximizes the coverage area, but increases the overlap area. Jin and Tang [38] developed a coverage path planning algorithm which respects the kinematic constraints of the robot, but it is specialized for agricultural or similar purposes and is computationally complex. Lee et al. [39] presented a smooth CCPP algorithm based on Bézier curve, wall following, and high-resolution grid map representation. The algorithm maximizes coverage rate and minimizes energy consumption, but has increased coverage redundancy. All the above smoothing algorithms are not suitable for changing environments due to high computational complexity.



**Figure 1.** Solution examples of different methods for the TSP problem. (a) The RSTC algorithm, (b) The PT algorithm, (c) The ACO algorithm, (d) The HNN algorithm.

### 3. The Proposed Smooth Complete Coverage Path Planning Algorithm

The pipeline of the SCCPP algorithm, shown in Figure 2, consists of four sequentially computed modules. The first module is the replanning spanning tree coverage algorithm (RSTC), which generates a path composed of straight-line path segments perpendicular to each other (Section 3.1). The algorithm initially requires the map of the environment and the starting pose of the robot. The initial map can be a CAD map or a map created with a simultaneous localization and mapping algorithm (SLAM). The second module is the path smoothing algorithm, which smooths the path generated by the RSTC algorithm using clothoids (Section 3.2). The third module is the velocity profile optimization algorithm, which computes the highest admissible velocity profile along the smoothed path accounting for kinematic and dynamic constraints of the robot (Section 3.3). The fourth module is the trajectory tracking algorithm, which ensures that the robot tracks the optimal 5D trajectory generated by the third module (Section 3.3).



**Figure 2.** The pipeline of the smooth complete coverage path planning algorithm.

#### 3.1. The Replanning Spanning Tree Coverage Algorithm

We use the replanning spanning tree coverage (RSTC) algorithm [10], which we briefly describe here. A map of the environment, given in the Portable Network Graphics (PNG) format, was edited to align the walls with the grid for better performance of the coverage algorithm and downsampled to a low-resolution occupancy grid with equally sized square cells (Figure 3a). Each cell contains occupancy information depending on the obstacle position. Only completely free cells were considered in the path planning and they are divided into four equal subcells of size  $D$ , where  $D$  is the diameter of the circumscribed circle around the robot's footprint (Figure 3b).

To create an optimal path, which visits each subcell exactly once, a spanning tree is constructed (Figure 3c). Free  $2D$ -size cells induce an undirected graph structure whose nodes are the centers of the cells, while the edges are the connecting lines between centers of the adjacent cells that share a common cell side. A spanning tree of an undirected graph is a subgraph that includes all nodes of the graph, with a minimum possible number of edges. A spanning tree is found in linear time by using the Breadth First Search (BFS) algorithm, meaning that the candidate neighbor nodes are kept in a queue. The pseudocode for spanning tree determination is given by Algorithm 1. The construction of the spanning tree starts from the node which contains the robot's position. All its neighbor nodes are inserted into the queue. The second node of the tree defines the first edge of the spanning tree, which has a direction aligned with the robot's orientation, if possible. The spanning tree continues to grow by removing from the queue and connecting to the last added node of the tree the node that is its neighbor. If this node has no neighbors in the queue, the node added one step earlier is compared to the queue, and so on. The tree is constructed when there are no nodes in the queue.

**Algorithm 1** Pseudocode for the spanning tree

---

```

1: Input: occupancyGridMap, currentPose
2: Output: spanningTree
3: insert starting cell which contains the robot's position in the queue spanningTree
4: while all free cells in occupancyGridMap have been visited do
5:   determine all orthogonal and unvisited neighbors of the current cell moving counter-clockwise and add them to the queue neighbor
6:   for all cells in queue neighbor do
7:     if distance of current cell in spanningTree and neighbor  $\leq 1$  then
8:       add current cell to the queue spanningTree
9:       set current cell in occupancyGridMap as visited
10:      go to the neighbor cell
11:     end if
12:   end for
13:   if distance of current cell in spanningTree and neighbor  $> 1$  then
14:     for cells in queue spanningTree do
15:       if distance of current cell in spanningTree and neighbor  $\leq 1$  then
16:         go to the neighbor cell
17:       else
18:         move to the previous cell in queue spanningTree
19:       end if
20:     end for
21:   end if
22: end while
23: return the spanning tree

```

---

Once the spanning tree is created, the coverage path computation begins. The pseudocode for the path planning is given by Algorithm 2. The robot is assumed to follow the path around the spanning tree, always on the right side, until it completely covers all subcells. The coverage ends when the robot gets to the start subcell. The result is the complete coverage path, which consists of a series of connected lines (Figure 3d).

**Algorithm 2** Pseudocode for the path planning

---

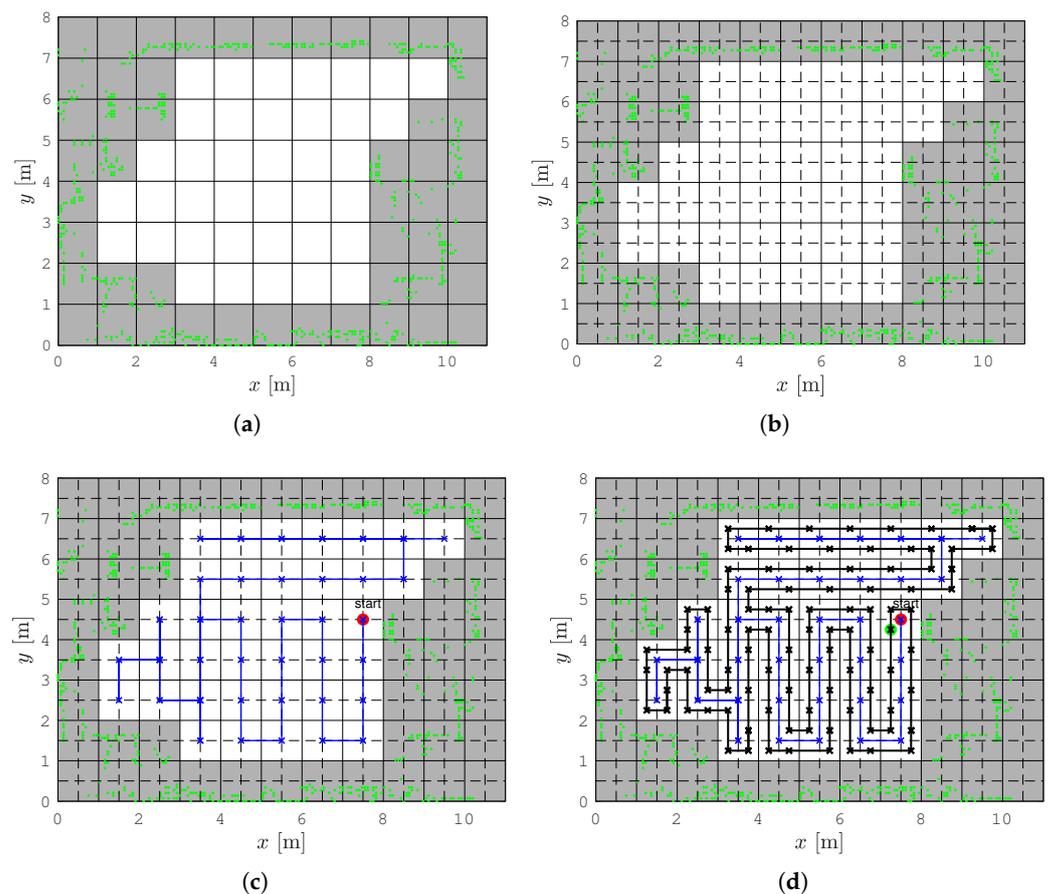
```

1: Input: spanningTree, currentPose
2: Output: RSTCpath
3: while it reaches the starting subcell again do
4:   calculate the direction of the spanning tree from current cell to the next first neighbor which is connected with the edge in the spanning tree
5:   mark current subcell as visited
6:   add subcell center coordinates in the queue RSTCpath
7:   go to neighbor cell
8:   if dynamic obstacle is detected then
9:     end the path planning algorithm
10:  end if
11: end while
12: return the RSTC path

```

---

Figure 1 shows a comparison of the RSTC algorithm with heuristic methods that solve TSP: the PT, ACO, and HNN algorithms. The length of the path for all methods is 76 m. The RSTC path has the smallest number of turning points (42 turning points compared to 48 for PT, 56 for ACO, and 50 for HNN). The higher number of turns results with longer coverage execution time. Therefore, our algorithm produces the coverage path more suitable for nonholonomic robots than other TSP solutions. The RSTC path has the lowest computational time (measured in Matlab/Python implementation) (0.32 s compared to 1.56 s for PT, 124 s for ACO, and 0.6 s for HNN).



**Figure 3.** Example of the complete coverage path planning for the known environment step-by-step. The real obstacles collected by SLAM algorithm are presented with green dots. (a) Low resolution occupancy grid map, (b) Subdividing of grid cells, (c) The spanning tree (blue line), (d) Complete coverage path (black line).

While the robot follows the planned path and visits subcells one by one, it may encounter an unknown obstacle (Algorithm 3). In such a case, the new spanning tree is created for the rest of the unvisited grid cells and the path is recomputed. The robot continues to follow the new path from the right side of the spanning tree until it returns to the cell where the replanning started. It then continues to follow the previously planned path. If all cells are visited then the shortest path around the obstacle is determined and connected with the previously planned path. If the unknown obstacles free occupied cells, set these cells as free in the occupancy grid map. Add these cells to the previously determined spanning tree if the current cell is the diagonal neighbor of the occupied cells. The new path around this spanning tree is determined. The complete coverage algorithm ends when the robot returns to the start subcell of the initial path. Since only completely free cells are considered, the coverage path is optimal and minimizes the overlap area.

In summary, the main advantages of the RSTC algorithm are that it generates the shortest path, minimizes the overlap area, and allows real-time path replanning in changing environments. However, its drawbacks are sharp turns of the planned path where a robot has to stop and reorient itself to continue, which is inefficient regarding the task duration and energy consumption.

**Algorithm 3** Pseudocode for the complete coverage path planning

---

```

1: Input: Map, currentPose
2: Output: RSTCpath
3: determine occupancy grid map based on png map image
4: determine spanning tree based on Algorithm 1
5: determine the RSTC path based on Algorithm 2
6: while it reaches the starting subcell again do
7:   if unknown obstacle is detected then
8:     occupy cells in which unknown obstacle is detected
9:     if exist the unvisited grid cells then
10:      determine the new spanning tree based on Algorithm 1 for the rest unvisited
grid cells
11:      determine the new RSTC path based on Algorithm 2 for the rest unvisited
grid cells
12:     else
13:      determine the RSTC path around the obstacle so that the minimum number
of double-covered subcells is obtained and connect it with previously planned path
14:     end if
15:   end if
16:   if unknown obstacle cleared occupied cells then
17:     set these cells as free in the occupancy grid map
18:     if current cell is diagonal neighbor of the occupied cell then
19:       add these cells in the previously determined spanning tree
20:       determine the new RSTC path based on Algorithm 2
21:     end if
22:   end if
23: end while
24: return the RSTC path

```

---

**3.2. Path Smoothing**

The task of the path smoothing algorithm is to smooth the path generated by the RSTC algorithm at the sharp turns to allow continuous motion of the robot without stopping. The path smoothing algorithm [11] is used, which uses clothoids to smoothly connect adjacent lines of the path.

The coordinates of a general clothoid are:

$$x(s) = x_0 + \int_0^s \cos(\theta_0 + \kappa_0 \xi + \frac{1}{2} c \xi^2) d\xi, \quad (1a)$$

$$y(s) = y_0 + \int_0^s \sin(\theta_0 + \kappa_0 \xi + \frac{1}{2} c \xi^2) d\xi, \quad (1b)$$

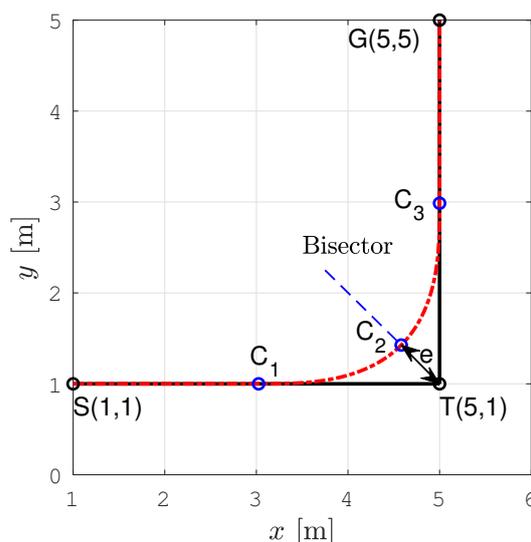
where  $(x_0, y_0, \theta_0)$  is the initial robot pose, the parameter  $\kappa_0$  is the initial curvature at  $s \geq 0$ , which denotes the arc length, and the parameter  $c$  is the sharpness, which describes how much the curvature changes with traveled distance. Curvature  $\kappa$  can be expressed as a linear function of the arc length  $s$ :

$$\kappa(s) = \kappa_0 + cs. \quad (2)$$

The Equation (1) contain Fresnel integrals, which are transcendental functions that cannot be solved analytically, making them difficult to use in real-time applications. To enable the use of clothoids in real-time, various methods have been developed and we use the one that is particularly fast and thus very suitable for real-time applications [11]. It is based on a so-called basic clothoid with bounded approximation error [40] whose

coordinates can be stored in a lookup table, and the coordinates of any other clothoid can be determined based on the points of the basic clothoid by rescaling, rotating, and translating.

An example of the smoothed sharp turn is shown in Figure 4, where the points  $S$ ,  $T$ , and  $G$  denote the start, sharp turn, and the goal position, respectively. The smooth path is shown as a red curve consisting of four parts: a straight line  $\overline{SC_1}$ , the first clothoid from  $C_1$  to  $C_2$ , the second clothoid from  $C_2$  to  $C_3$ , and a straight line  $\overline{C_3G}$ . The two clothoids are symmetric—the second one is obtained by mirroring the first one at the bisector of the angle spanned by the lines  $\overline{ST}$  and  $\overline{TG}$ .



**Figure 4.** Illustration of the sharp turn smoothing by two clothoids.

The parameter  $e$  is the deviation from the planned path. If it is high, the path curvature is low and the robot can drive at a higher velocity. Since for our problem, each sharp turn is at a  $90^\circ$  angle, the same deviation parameter  $e$ , and consequently the same pair of symmetric clothoids can be applied for all of them, which additionally simplifies the smoothing.

We determined the maximum deviation  $e_{\max}$  to ensure that the robot's footprint does not collide with the obstacle cell at the point of maximal deviation, i.e.:

$$e_{\max} \leq \frac{D\sqrt{2} - D}{2}, \quad (3)$$

where  $D\sqrt{2}$  is the subcell diagonal, and  $D$  is the diameter of the circumscribed circle around the robot's footprint. In our setup it is  $D = 0.5$  m, so we choose  $e_{\max} = 0.1$  m. From  $e_{\max}$  it is possible to determine the maximum curvature in point  $C_2$  (see [11]), so we obtain  $\kappa_{\max} = 6.4 \text{ m}^{-1}$ .

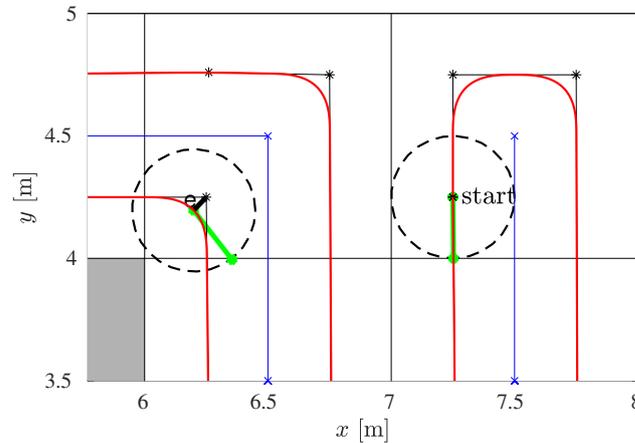
From the radial acceleration, which is  $a_{\text{radial}} = v \times \omega = v^2 \kappa$ , where  $v$  and  $\omega$  are the linear and angular velocity of the robot, respectively, we can get the velocity limit at the curvature extrema as:

$$v_{\min} = \sqrt{\frac{a_{\text{radial}_{\max}}}{\kappa_{\max}}}. \quad (4)$$

Since  $a_{\text{radial}_{\max}}$  is characteristic of the mobile robot, and in our setup it is  $a_{\text{radial}_{\max}} = 0.1 \text{ m/s}^2$ , this gives the velocity limit  $v_{\min} = 0.125 \text{ m/s}$ , which is used later in the velocity profile optimization module.

Figure 5 shows a part of the original and smoothed path with two positions of the robot. It can be noticed that the robot's circular footprint when moving along the smoothed path is always within the free subcells of size  $D$ . It does not collide with the obstacle cell even at points where a deviation of the smoothed path from the original path is the largest (noted with  $e$ ).

As described in Section 3.1, the RSTC algorithm recalculates the path for the unvisited grid subcells when the robot encounters an unknown obstacle in the neighboring cells. Due to the computational simplicity of the proposed path smoothing algorithm, the smoothing of the replanned path can be conducted online so that the robot does not need to slow down or stop.



**Figure 5.** The enlarged part of the coverage path with two positions of the robot: (1) the robot's starting position and (2) the robot's position at the obstacle corner. The obstacle cell is shown in gray, the robot's positions in dashed circles, the robot's orientation in green, the spanning tree in blue, the RSTC path in black, and the smoothed path in red.

### 3.3. Velocity Profile Optimization and Trajectory Tracking

Although the path smoothing algorithm takes into account the kinematic and dynamic constraints of the robot, it only outputs  $x$  and  $y$  coordinates and curvature of the smooth complete coverage path. To ensure time-optimal robot motion along the smooth path, the velocity profile with maximum allowable velocities should be calculated. The turning points of the smoothed path (e.g., as point  $C_2$  in Figure 4) are the local extrema points of the smooth path curvature, i.e., at these points, the turning radius reaches a local minimum, and consequently, the velocity is locally lowest. The maximum allowable velocity of the robot at a turning point is determined by the radial acceleration limit  $a_{\text{radial}}$ . The robot must decelerate and accelerate before and after the turning point as much as maximally tangentially allowed by the acceleration limit. The robot should always remain within its acceleration limits. The overall acceleration is calculated as:

$$a = \sqrt{a_{\text{radial}}^2 + a_{\text{trans}}^2} \quad (5)$$

where  $a_{\text{radial}} = v^2\kappa$  is the radial acceleration of the robot and  $a_{\text{trans}} = \frac{dv}{dt}$  is the translation acceleration.

We used the velocity profile optimization algorithm described in detail in [41]. It outputs the optimal linear and angular velocities of the robot based on the points of the smooth complete coverage path and the driving limitations. Linear and angular velocities, together with the robot position and orientation, result in the 5D trajectory  $(x, y, \theta, v, \omega)^T$ . This trajectory is the input to the trajectory tracking algorithm based on the Kanayama controller [42]. This controller ensures that the robot tracks the optimal 5D trajectory by dumping perturbations of the robot pose provided by the localization module and velocity measurements from the wheel encoders.

The kinematic model of the differential drive mobile robot can be represented as follows:

$$\begin{bmatrix} \dot{x}(t) \\ \dot{y}(t) \\ \dot{\theta}(t) \end{bmatrix} = \begin{bmatrix} \cos\theta(t) & 0 \\ \sin\theta(t) & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v(t) \\ \omega(t) \end{bmatrix}, \quad (6)$$

where  $t$  is time,  $(x(t), y(t))$  is position,  $\theta(t)$  is orientation,  $v(t)$  is linear velocity, and  $\omega(t)$  is the angular velocity of the mobile robot. For the system (6), the nonholonomic constraint is:

$$-\dot{x}(t) \cdot \sin\theta(t) + \dot{y}(t) \cdot \cos\theta(t) = 0, \quad (7)$$

which constrains the drive wheels to roll and prevents slippage. A path is feasible for nonholonomic mobile robot if it can follow the path with the reference velocity commands.

The main task of the tracking control for mobile robots is to find appropriate velocities  $v(t)$  and  $\omega(t)$  to achieve the control objective. Inputs for the Kanayama controller are the reference configuration  $\mathbf{p}_r = (x_r, y_r, \theta_r)^T$ , which is taken from the 5D trajectory at the current time instant, and the measured current configuration  $\mathbf{p}_c = (x_c, y_c, \theta_c)^T$ . The Kanayama controller outputs are the corrected reference linear and angular velocities. The first step of the control law is to compute the error configuration  $\mathbf{p}_e$  as the difference between  $\mathbf{p}_r$  and  $\mathbf{p}_c$ , which must converge to zero. For the kinematic model given by (6) and (7), the expression for the error configuration is:

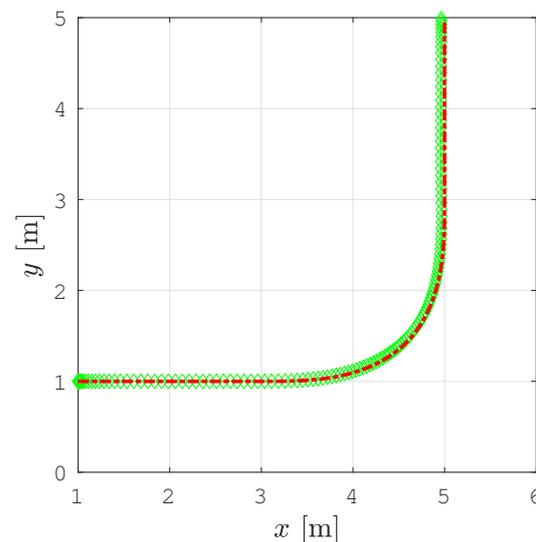
$$\mathbf{p}_e = \begin{bmatrix} x_e \\ y_e \\ \theta_e \end{bmatrix} = \begin{bmatrix} \cos\theta_c & \sin\theta_c & 0 \\ -\sin\theta_c & \cos\theta_c & 0 \\ 0 & 0 & 1 \end{bmatrix} (\mathbf{p}_r - \mathbf{p}_c). \quad (8)$$

Using the error configuration  $\mathbf{p}_e$ , the reference linear and angular velocities  $v_r$  and  $\omega_r$ , the corrected linear and angular velocities  $v$  and  $\omega$  can be calculated as follows:

$$\begin{bmatrix} v \\ w \end{bmatrix} = \begin{bmatrix} v_r \cos\theta_e + K_x x_e \\ \omega_r + v_r (K_y y_e + K_\theta \sin\theta_e) \end{bmatrix}, \quad (9)$$

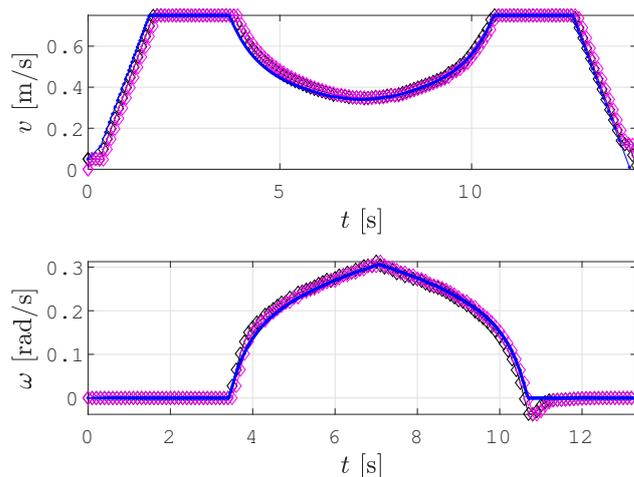
where  $K_x$ ,  $K_y$  and  $K_\theta$  are the positive constant parameters of the controller.

Figure 6 shows the smoothed path and the tracked trajectory for the example shown in Figure 4. There is no significant deviation of the tracked trajectory from the smoothed path and the mobile robot moves smoothly from one segment of the path to another.



**Figure 6.** The smoothed path from Figure 4 (red line) and the tracked trajectory (green diamonds).

Figure 7 shows linear and angular velocities for the example shown in Figure 4 calculated by the velocity profile optimization procedure (blue line) and the real linear and angular velocity profile, while the robot tracked the trajectory (magenta diamonds). The corrected reference velocities calculated by Kanayama's algorithm are shown with black diamonds. The robot in the simulation reached exactly the corrected reference velocity with a delay of one control time step.



**Figure 7.** The calculated linear and angular velocities (blue line) and real linear and angular velocities (magenta diamonds) at which the robot tracked the trajectory.

#### 4. Simulation Results

We tested the proposed motion planning approach in three scenarios: the *Lab scenario* (Section 4.1), *Aula scenario* (Section 4.2), and *Gallery scenario* (Section 4.3). For each scenario, the proposed SCCPP algorithm is compared with the original RSTC algorithm without smoothing (called the CCPP algorithm in the following), and the results are discussed in Section 4.4. We also compared SCCPP with CCD\* and HDCP algorithms for the *Gallery scenario*. CCD\* was selected for comparison because it is a graph search method based on D\* search of the high resolution grid map of the environment. HDCP was selected for comparison because it plans smooth paths based on Dubins curve, which is followed at constant velocity in real-time.

We used a receding horizon control (RHC) algorithm developed within our research group [43] for the path following of the the CCPP without smoothing. This algorithm can adapt to dynamic changes in the environment. If the selected subgoal is too close to the detected obstacle, a new subgoal is chosen in the local vicinity from the critical one. With this procedure, the robot follows the planned path with a minimal drift.

The simulations were performed on an Asus ROG Strix SCAR II (Intel i7-8750H, 16 GB DDR4 RAM). The algorithms were implemented and tested in the Robot Operating System (ROS). The stage simulator was used for the simulations.

To create the environment map, for each test scenario the environment was explored through sensors on the Pioneer 3DX robot and data was collected by the *slam\_gmapping* ROS package. Then, the map was created and post-processed to align walls to the grid for better performance of the coverage algorithm. The post-processed map is resampled to an occupancy grid map with low resolution. The size of the cells is  $1 \times 1$  m and each cell is divided into four  $0.5 \times 0.5$  m sub-cells, which are approximately of the same size as the robot's footprint. The *amcl* ROS package was used for the robot localization in created maps.

The *Lab scenario* tests how the algorithm works in a small room with static and dynamic obstacles. The *Aula scenario* tests how the algorithm works in a large hallways with a large number of static obstacles. The third scenario is the *Gallery scenario*, which tests the algorithm performances in narrow spaces. To validate the performances of the SCCPP algorithm and to compare it with other algorithms, the path length and execution time were measured, and the coverage rate and coverage redundancy were calculated.

The coverage rate is calculated as:

$$\text{Coverage rate} = A_p / (A_t - A_o) \cdot 100\%, \quad (10)$$

where  $A_t$  is the total area of the environment,  $A_o$  is the area of obstacles, and  $A_p$  is the area covered by the driven trajectory of the robot. The coverage redundancy is calculated as follows:

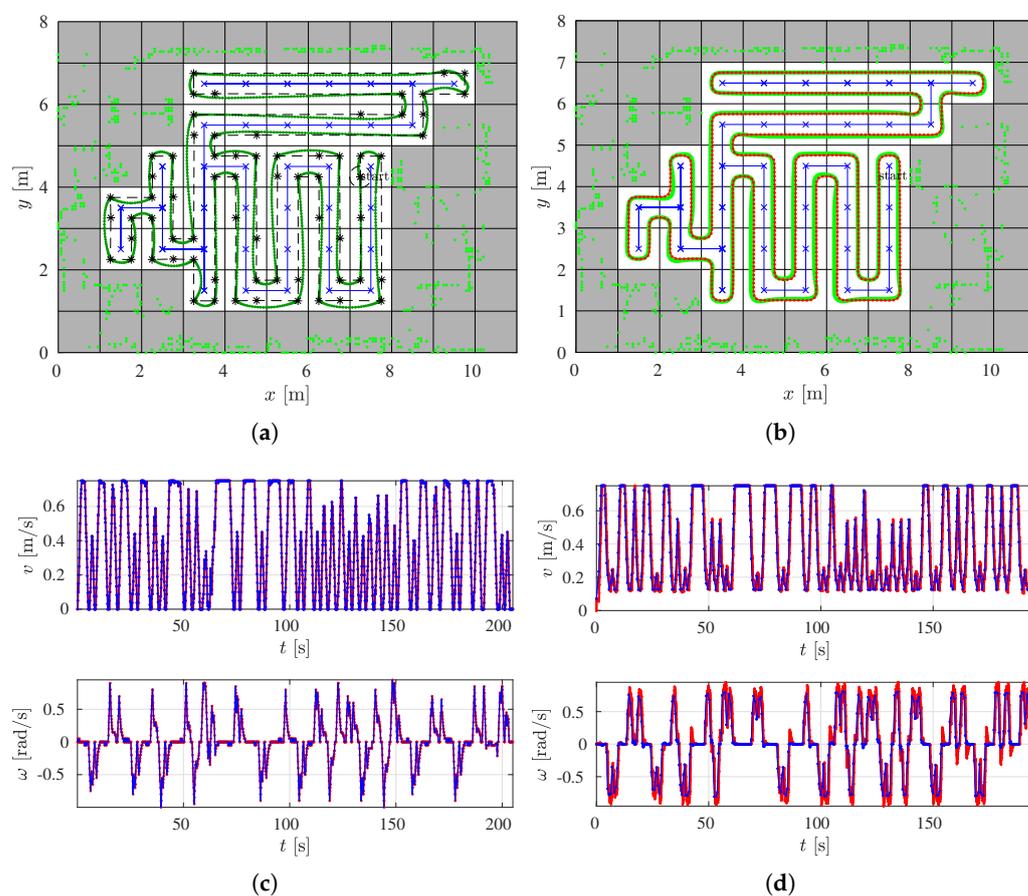
$$\text{Coverage redundancy} = A_r/A_p \cdot 100\%, \quad (11)$$

where  $A_r$  is the overlap area, calculated as the sum of all subcells visited more than once.

In all figures below that show the paths or trajectories, static obstacles are shown with green dots, and if they partially occupy a cell, the entire cell is shown as occupied (gray cells), the spanning tree is shown as a blue line, the RSTC path as a black line, the smoothed RSTC path as a red line, and the tracked trajectory as a green line.

#### 4.1. The Lab Scenario

The coverage starts at cell (7, 4) which is the starting cell for the spanning tree construction and the path circumnavigates around the constructed spanning tree (see Figure 8a). The robot follows the path from the right side of the spanning tree and covers each subcell exactly once. The coverage is complete when the robot returns to the start subcell.



**Figure 8.** The comparison of the CCPP and SCCPP algorithms in the *Lab scenario* with the presented RSTC spanning tree (blue line), the RSTC path (black line), the smoothed path by the SCCPP algorithm (red line), the executed path by the CCPP and SCCPP algorithm (green line), the reference velocity profile (blue line), and the actual velocity profile (red line). (a) The CCPP path in the *Lab scenario*. (b) The SCCPP path in the *Lab scenario*. (c) The CCPP velocity profile in the *Lab scenario*. (d) The SCCPP velocity profile in the *Lab scenario*.

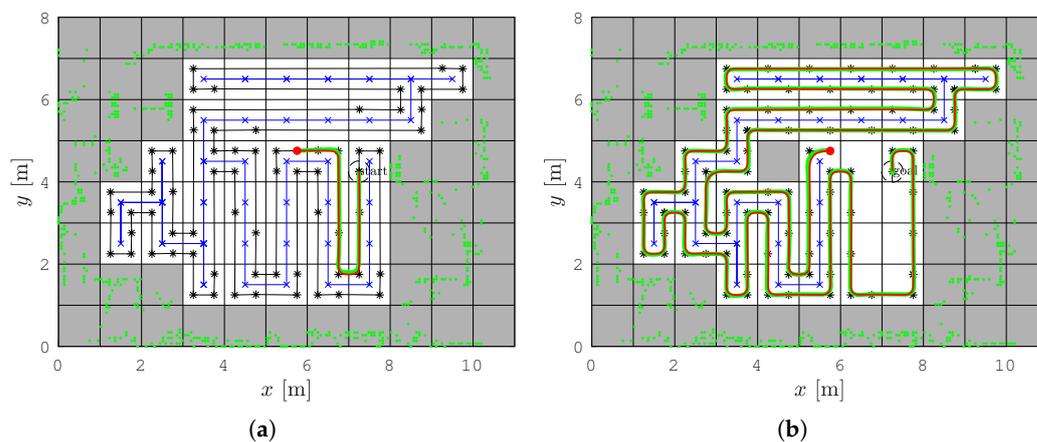
The path is created with straight lines that form sharp turns. In each turn for the CCPP variant, the robot must stop and reorient itself so that the robot heading is equal to the path direction. For this reason, the linear velocity is zero and the angular velocity is close to the maximal value. On straight sections of the path, the linear velocity is maximal and

the angular velocity is zero. Figure 8c shows the calculated linear and angular velocities (blue line) and the real linear and angular velocities of the robot executed by the motors (red line).

To improve the coverage and reduce the execution time, the smoothed variant—the SCCPP algorithm is used. The output of this algorithm is the smoothed path that circumnavigates around the constructed spanning tree (see Figure 8b). In this way, sharp turns are now smoothed and the robot can travel without stopping.

The execution of the SCCPP algorithm can be examined from the linear and angular velocities shown in Figure 8d. The blue lines are the linear and angular velocities calculated by the optimization process, and the red lines show how the linear and angular velocities changed as the robot tracked the trajectory. It can be observed that the linear velocity is not zero while the robot has to overcome the turn as opposed to zero velocities in Figure 8c, where the smoothing algorithm is not used.

The replanning SCCPP algorithm is executed in a dynamic environment. Dynamic changes can be detected by the robot in neighboring cells of the current cell where the robot is currently located; see Figure 9a for an example where the obstacle is detected in cell (4, 4). All unvisited cells are considered as nodes for the new spanning tree calculation and the visited subcells are not part of the new path coverage, see Figure 9b. After the new spanning tree is determined for the rest of the unvisited grid cells, the new path is determined. The path always follows the spanning tree counterclockwise. When the robot returns to the cell where the replanning started, it continues to follow the path from the right side of the spanning tree that was already formed during the previous path planning. Therefore, this part of the spanning tree is not shown in Figure 9b from cell (4, 4) to starting cell (7, 5). The robot traverses these visited grid cells by visiting only subcells that are not yet visited. The complete coverage algorithm ends when the robot gets to the starting subcell. The red dots in Figure 9 are the position of the robot where the replanning algorithm is executed. In this example, the replanning process is fast enough that the robot does not need to stop for the recalculation of the smoothed path. The recalculation of the smoothed RSTC path takes less than 1 ms on average.

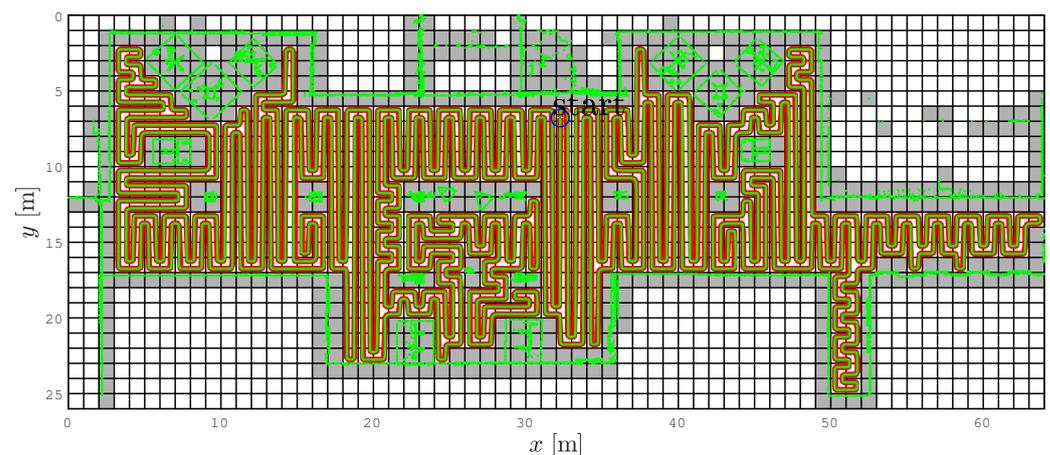


**Figure 9.** The RSTC algorithm in the *Lab scenario* with the presented RSTC spanning tree (blue line), the RSTC path (black line), the smoothed path by the SCCPP algorithm (red line), and the executed path by the smoothed RSTC algorithm (green line). Red points are the robots' position where the replanning algorithm is executed. (a) The smoothed RSTC path in the *Lab scenario* before replanning. (b) The smoothed RSTC path in the *Lab scenario* after replanning.

#### 4.2. The Aula Scenario

Similar behavior of the CCPP and SCCPP algorithms can be observed in the example with few hallways and more static obstacles; see Figure 10. Static obstacles are shown with green dots and if they partially or fully occupy a cell, the cell is presented as occupied (gray cells). Some static obstacles are detected by the laser outside the building boundaries

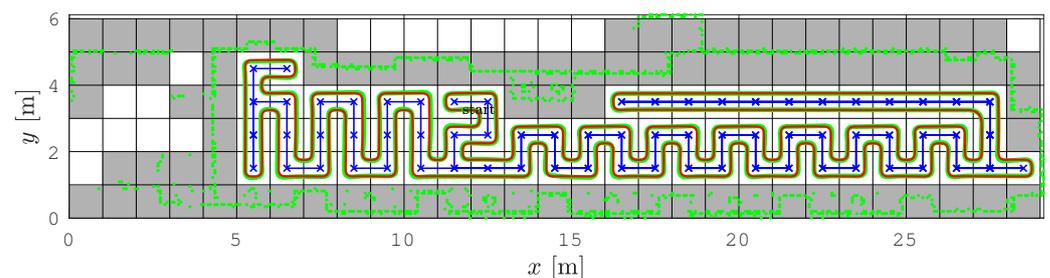
because there are windows in parts of the hallway and these grid cells are represented as occupied. The direction of the first segment of the complete coverage path is  $-90^\circ$  due to the initial orientation of the robot. The coverage starts at cell (6, 32), which is the starting cell for the spanning tree construction and the path circumnavigates around the constructed spanning tree, but this is not represented in Figure 10. Due to the transparency of the figure we presented, only a smoothed path (red line) and driven trajectory (green line). The smoothed path does not contain sharp turns and the robot can drive the trajectory without stopping and reorienting itself to align heading with the path direction. This improves the coverage rate and reduces the execution time of the coverage task. The SCCPP variant outperforms the CCP algorithm here as well.



**Figure 10.** The SCCPP algorithm in the *Aula scenario*, which presents the smoothed path with red line and the executed path with green line.

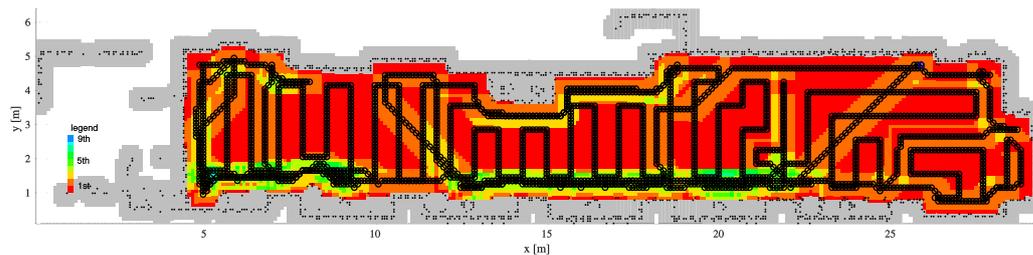
#### 4.3. The Gallery Scenario

This scenario has the most turns due to the narrow dimensions of the environment. The starting cell for the spanning tree construction and the RSTC path is (11, 3), see Figure 11. Again, the SCCPP algorithm variant is faster than the CCP variant.



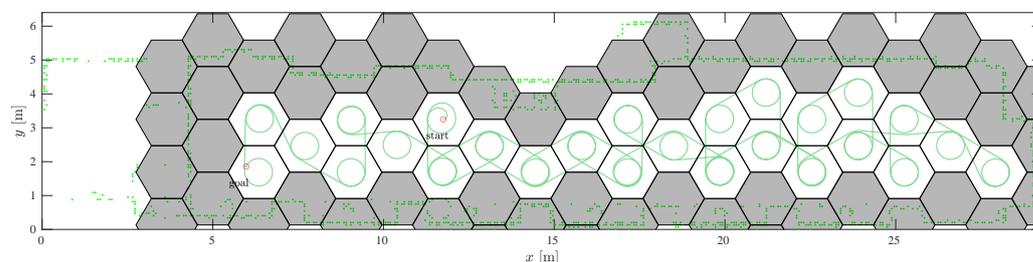
**Figure 11.** The SCCPP algorithm in the *Gallery scenario*: the RSTC spanning tree (blue line), the smoothed path (red line), and the executed path (green line).

The SCCPP algorithm is compared to the CCD\* and HDCP algorithms in this scenario. The complete coverage path of the CCD\* algorithm is shown in Figure 12 with the black line, and visited cells are marked by different colors according to the number of visits, so some parts of the map are visited nine times.



**Figure 12.** The complete coverage of the CCD\* algorithm with the path and noted redundant numbers of cell visits (from 1 to 9).

The complete coverage path of the HDCP algorithm is shown in Figure 13 with the green line. Partially occupied cells are shown in gray hexagons and only completely free hexagonal cells are used for the coverage task. The path begins in the cell (11, 3) and ends in the cell (6, 2). Some cells are visited more than once due to the narrow environment and low resolution occupancy grid map.



**Figure 13.** The complete coverage of the HDCP algorithm with hexagonal cell grid.

4.4. Discussion

The results of the CCPP and SCCPP comparison in all three scenarios are given in Table 1. When the smoothing algorithm is used, the length of the tracked trajectory is shorter and the execution time is reduced compared to the results obtained without smoothing. In the *Gallery scenario*, the RSTC algorithm provides a path with many sharp turns and because of the discontinuity during the motion, the execution time for the complete coverage is about 10% longer for CCPP than for SCCPP. It can be noted that the environment configuration and the position of the obstacles can affect the execution time and the total length of the tracked trajectory. In addition, the CCPP algorithm has higher localization errors at points where the robot rotates in place. These cause higher uncertainty in the robot’s pose and the deviation from the tracked trajectory is larger than in the SCCPP examples. This also leads to an increase in the coverage redundancy.

**Table 1.** Table of comparison of CCPP and SCCPP algorithms.

	Lab CCPP	Lab SCCPP	Gallery CCPP	Gallery SCCPP	Aula CCPP	Aula SCCPP
Coverage path length	77.22 m	<b>72.46 m</b>	141.04 m	<b>129.44 m</b>	1255.91 m	<b>1191.22 m</b>
Coverage time	217.30 s	<b>193.26 s</b>	453.99 s	<b>409.58 s</b>	3479.51 s	<b>2834.23 s</b>
Coverage rate	71.48%	<b>74.42%</b>	66.37%	<b>72.11%</b>	79.03%	<b>79.84%</b>
Coverage redundancy	35.53%	<b>7.89%</b>	48.19%	<b>5.79%</b>	39.23%	<b>3.37%</b>
Nodes number	61	61	97	97	744	744
Path calculation time	<b>0.5 ms</b>	0.9 ms	<b>0.6 ms</b>	1.2 ms	<b>5 ms</b>	10 ms

From these three scenarios, it can be observed that the SCCPP algorithm has, on average, a 6.5% shorter tracked trajectory, an 8.8% reduction in coverage execution time, a 4.5% better coverage rate, and an 82.34% lower coverage redundancy than the original CCPP algorithm. The better coverage rate of the SCCPP compared to the CCPP can be explained by higher path following accuracy, while accuracy of the CCPP is worsened by sharp turns. To provide the scalability of the proposed algorithm, each scenario has

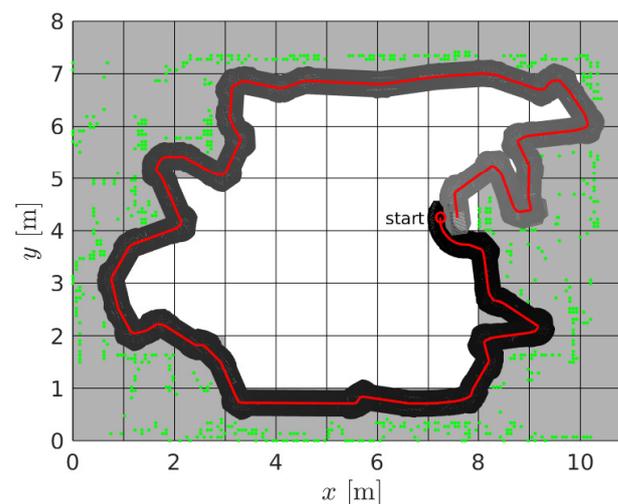
a different size of the environment, which leads to different number of nodes (cells) in a spanning tree construction. Because the number of nodes is proportional to the computation time, *Lab scenario* has the smallest and *Aula scenario* has the biggest elapsed time required to calculate the complete coverage path. For each scenario, SCCPP has a larger time than CCPP for the time required to calculate the smooth path. The time increases linearly with the number of nodes, while the number of nodes increases quadratically with the map dimensions.

The SCCPP algorithm is compared to the CCD\* and HDCP algorithms, and the results are shown in Table 2. Compared to CCD\* and HDCP, SCCPP has the shortest coverage path length (42% shorter than CCD\* and 21% shorter than HDCP), the shortest coverage time (47% shorter than CCD\* and 53% shorter than HDCP), and the smallest coverage redundancy (82% smaller than CCD\* and 40% smaller than HDCP). However, SCCPP has worse coverage rate compared to CCD\* (27% lower), and better coverage rate compared to HDCP (15% higher). The reason of the lower coverage rate of both the SCCPP and HDCP algorithm is the used low resolution occupancy grid map. However, the high resolution occupancy grid map results with higher coverage redundancy in CCD\*.

**Table 2.** Comparisons of the SCCPP, CCD\* and HDCP algorithms for the *Gallery scenario*.

	SCCPP	CCD*	HDCP
Coverage path length	<b>129.44 m</b>	222.79 m	163.48 m
Coverage time	<b>409.58 s</b>	768.22 s	879.65 s
Coverage rate	72.11%	<b>98.85%</b>	57.38%
Coverage redundancy	<b>5.79%</b>	87.48%	46.15%

The coverage rate for the SCCPP algorithm can be increased if a wall following method is used, but this also increases the redundancy. The wall following algorithm used after SCCPP is presented in Figure 14 with driven trajectory (red line) and robot's footprint at every point on the trajectory (degradation from black to gray). The coverage rate is increased to 97.64% with the use of the wall following, and the coverage redundancy is increased to 24.64%.



**Figure 14.** The wall following algorithm after SCCPP: driven trajectory (red line), and robot's footprint in every point on the trajectory is presented degraded from black to gray.

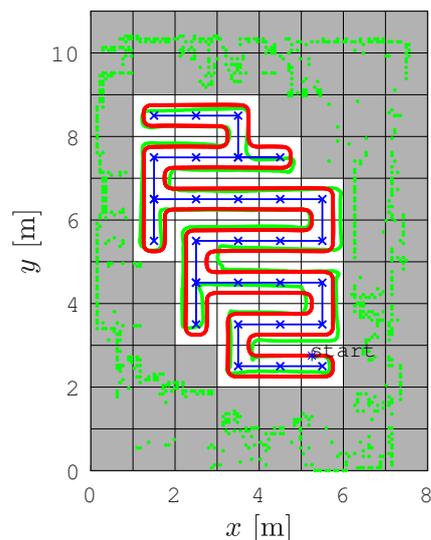
The advantages of our SCCPP algorithm are completeness of coverage, robustness to environmental shape and initial robot pose, optimal path that visits all subcells exactly once, time efficiency, low coverage redundancy, and fast replanning. The coverage rate can be significantly increased if the wall following method is used. The limitation is that the algorithm requires a priori knowledge about the workspace. The execution of the proposed

smooth complete coverage path planning algorithm is around 10 ms and it is suitable for real-time operation due to its computational simplicity.

### 5. Experiments on a Real Robot

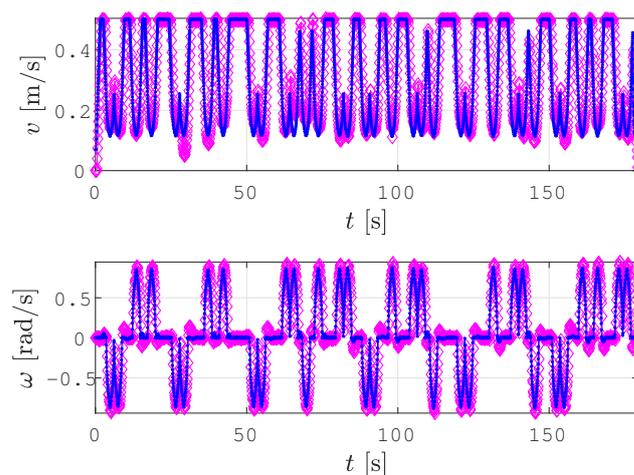
The experiment (the experiments are demonstrated in the accompanying video available here: <https://youtu.be/VEAGHIAIpRA> accessed on 18 October 2022) was performed on the Pioneer 3DX robot with a SICK laser sensor LMS200. To have a similar scenario as in the simulation, the experiment was performed in the *Lab scenario*, where a new map had to be created due to some minor changes in the placement of the furniture. First, the *Lab* was explored using the laser sensor on the robot, and the data of the environment was collected using the *slam\_gmapping* ROS package. After the environment was explored, the map was created and the package *amcl* ROS was used for robot localization in the created map.

The SCCPP algorithm was executed and the results are shown in Figure 15. The static obstacle configuration is represented as green dots, which are shown as occupied cells (gray cells in Figure 15) in the occupancy grid map. The initial position of the robot ( $x = 5.25$  [m],  $y = 2.75$  [m]) is marked with a blue star and this is the starting position for the coverage of the environment. The blue line is the spanning tree determined by RSTC algorithm, the red line is the smoothed RSTC path, and the green line is the driven trajectory by the Pioneer 3DX robot.

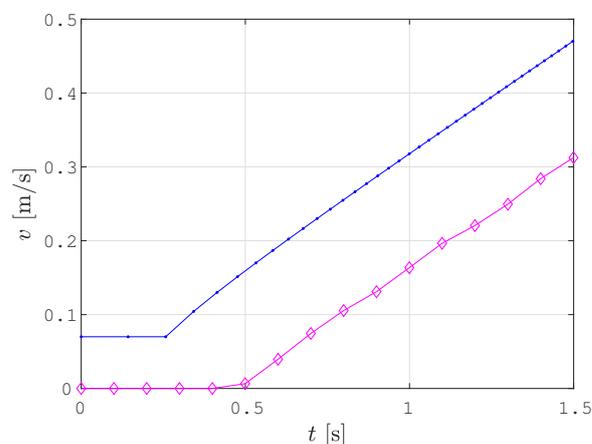


**Figure 15.** The calculated SCCPP path (red line) and trajectory tracked by the Pioneer 3DX robot (green line).

There are some deviations between the smoothed path determined by the SCCPP algorithm (red line) and the real trajectory tracked by the robot (green line), especially when dealing with parts of the path that are curved. This problem is due to the inaccurate and noisy localization of the robot. Another problem is the hardware setup. We used serial communication between the robot and the laptop with ROS, which caused a delay of three cycles in sending the calculated velocities to the robot. In Figure 16, the calculated linear and angular reference velocity profile are shown with blue lines and the actual linear and angular velocity profile of the robot during trajectory tracking is shown with magenta lines. For the experiment with the real robot, we used 0.5 m/s and 0.75 rad/s as the maximal linear and angular velocities, respectively. The delay of three cycles caused the corrected reference velocity by the Kanayama tracking algorithm to deviate from the optimized velocity profile of the smoothed path. This is shown in Figure 17, where the first 1.5 s of the linear velocity profile from Figure 16 are shown magnified.



**Figure 16.** Calculated linear and angular velocities (blue line) and linear and angular velocities, while the Pioneer 3DX robot is tracking the trajectory (magenta line).



**Figure 17.** Real linear velocity, while the robot Pioneer 3DX is following the trajectory (magenta line), deviates from the applied linear velocity (blue line).

Table 3 presents the experimental evaluation of the SCCPP algorithm efficiency on the Pioneer 3DX robot for the *Lab scenario*. The coverage path length is 53.85 m, the coverage execution time is 179.5 s, the coverage rate is 77.7%, and the coverage redundancy is 12.96%. The trajectory tracking error is determined as the difference of the calculated trajectory (Figure 15 red dotted line) and tracked trajectory (Figure 15 green line) and it is 6.43 m<sup>2</sup>. When compared with simulation results in Table 1 for the *Lab scenario*, although the maps are slightly different, it can be observed that the coverage rate in both cases is approximately equal, which confirms that our algorithm is equally efficient in the real robot setup. The coverage redundancy is slightly worse on a real robot, which is mainly caused by noisy localization and delay in the control loop.

**Table 3.** SCCPP algorithm on the Pioneer 3DX robot for the *Lab scenario*.

	SCCPP
Coverage path length	53.85 m
Coverage time	179.5 s
Coverage rate	77.7%
Coverage redundancy	12.96%
Tracking error	6.43 m <sup>2</sup>

## 6. Conclusions

The proposed SCCPP algorithm is the online algorithm that generates a traversable collision-free trajectory based on clothoids with low computational cost. Such a path is suitable and feasible for nonholonomic mobile robots since it does not contain sharp turns. By using a smoothing technique on the proposed coverage path, the coverage efficiency can be significantly improved in terms of the time required and energy consumption during the coverage tasks and has very low overlap redundancy. The complexity of the environment affects the coverage efficiency, and the experimental results evaluated the efficiency of the CCPP algorithms on maps with different complexity levels. By using the SCCPP algorithm, the trajectory followed by the robot can be executed faster and with higher accuracy than without the smoothing algorithm. The SCCPP algorithm produces the shortest coverage path, takes the shortest time for coverage execution, and has the smallest coverage redundancy compared to the CCD\* and HDCCP algorithms. The SCCPP algorithm takes advantage of the large size of the grid cells to ensure real-time operation and minimization of overlapping areas, but at the cost of a lower coverage rate due to the uncovered areas around obstacles and walls. However, the coverage rate could be easily increased by simply combining our SCCPP algorithm with a wall following algorithm.

As future work, more experiments are planned for other robot designs such as omnidirectional mobile robots and Ackermann steering vehicles. Furthermore, we consider the extension of this work to multiple robots in the form of a decentralized solution for the coordinated multi-robot complete coverage task. This will decrease the total task time significantly due to the division of workload overall robots, while decentralization will prevent a single point of failure.

**Author Contributions:** Conceptualisation, A.Š., M.S., M.B. and I.P.; methodology, A.Š., M.S. and M.B.; software, A.Š. and M.B.; validation, A.Š.; formal analysis, A.Š.; investigation, A.Š.; resources, A.Š. and M.B.; data curation, A.Š.; writing—original draft preparation, A.Š., M.S., M.B. and I.P.; writing—review and editing, A.Š., M.S. and M.B.; visualisation, A.Š.; supervision, M.S., M.B. and I.P.; project administration, M.S. and I.P.; funding acquisition, I.P. All authors have read and agreed to the published version of the manuscript.

**Funding:** This work has been supported by the European Regional Development Fund under the grant KK.01.2.1.01.0138—Development of a multi-functional anti-terrorism system.

**Institutional Review Board Statement:** Not applicable.

**Informed Consent Statement:** Not applicable.

**Data Availability Statement:** Not applicable.

**Conflicts of Interest:** The authors declare no conflict of interest.

## References

1. Gao, X.S.; Xu, D.G.; Wang, Y. Omni-directional mobile robot for floor cleaning. *Chin. J. Mech. Eng.* **2008**, *44*, 228–233. [[CrossRef](#)]
2. Dakulović, M.; Horvatić, S.; Petrović, I. Complete Coverage D\* Algorithm for Path Planning of a Floor-Cleaning Mobile Robot. In Proceedings of the Preprints of the 18th IFAC World Congress, Milano, Italy, 28 August–2 September 2011; pp. 5950–5955. [[CrossRef](#)]
3. Yang, S.X.; Luo, C. A neural network approach to complete coverage path planning. *IEEE Trans. Syst. Man Cybern. Part B (Cybern.)* **2004**, *34*, 718–724. [[CrossRef](#)] [[PubMed](#)]
4. Acar, E.; Choset, H.; Zhang, Y.; Schervish, M. Path Planning for Robotic Demining: Robust Sensor-Based Coverage of Unstructured Environments and Probabilistic Methods. *Int. J. Robot. Res.* **2003**, *22*, 441–466. [[CrossRef](#)]
5. Dakulović, M.; Petrović, I. Complete coverage path planning of mobile robots for humanitarian demining. *Ind. Robot. Int. J.* **2012**, *39*, 484–493. [[CrossRef](#)]
6. Ollis, M.; Stentz, A. First results in vision-based crop line tracking. In Proceedings of the IEEE International Conference on Robotics and Automation, Minneapolis, MN, USA, 22–28 April 1996; pp. 951–956. [[CrossRef](#)]
7. Weiss-Cohen, M.; Sirotin, I.; Rave, E. Lawn Mowing System for Known Areas. In Proceedings of the 2008 International Conference on Computational Intelligence for Modelling Control & Automation, Vienna, Austria, 10–12 December 2008; pp. 539–544. [[CrossRef](#)]

8. Kapoutsis, A.C.; Chatzichristofis, S.A.; Doitsidis, L.; de Sousa, J.B.; Kosmatopoulos, E.B. Autonomous navigation of teams of Unmanned Aerial or Underwater Vehicles for exploration of unknown static dynamic environments. In Proceedings of the 21st Mediterranean Conference on Control and Automation, Platania, Greece, 25–28 June 2013; pp. 1181–1188. [\[CrossRef\]](#)
9. Kapoutsis, A.; Chatzichristofis, S.; Doitsidis, L.; Sousa, J.; Pinto, J.; Braga, J.; Kosmatopoulos, E. Real-time adaptive multi-robot exploration with application to underwater map construction. *Auton. Robot.* **2015**, *40*, 987–1015. [\[CrossRef\]](#)
10. Šelek, A.; Seder, M.; Petrović, I. Mobile robot navigation for complete coverage of an environment. *IFAC-PapersOnLine* **2018**, *51*, 512–517. [\[CrossRef\]](#)
11. Brezak, M.; Petrović, I. Path smoothing using clothoids for differential drive mobile robots. *IFAC Proc. Vol.* **2011**, *44*, 1133–1138. [\[CrossRef\]](#)
12. Cao, Z.L.; Huang, Y.; Hall, E.L. Region filling operations with random obstacle avoidance for mobile robotics. *J. Robot. Syst.* **1988**, *5*, 87–102. [\[CrossRef\]](#)
13. Galceran, E.; Carreras, M. A Survey on Coverage Path Planning for Robotics. *Robot. Auton. Syst.* **2013**, *61*, 1258–1276. [\[CrossRef\]](#)
14. An, V.; Qu, Z.; Roberts, R. A Rainbow Coverage Path Planning for a Patrolling Mobile Robot With Circular Sensing Range. *IEEE Trans. Syst. Man Cybern. Syst.* **2017**, *48*, 1238–1254. [\[CrossRef\]](#)
15. An, V.; Qu, Z.; Crosby, F.; Roberts, R.; An, V. A Triangulation-Based Coverage Path Planning. *IEEE Trans. Syst. Man Cybern. Syst.* **2020**, *50*, 2157–2169. [\[CrossRef\]](#)
16. Zelinsky, A.; Jarvis, R.; Byrne, J.C.; Yuta, S. Planning Paths of Complete Coverage of an Unstructured Environment by a Mobile Robot. In Proceedings of the International Conference on Advanced Robotics, Tokyo, Japan, 26–30 July 1993; pp. 533–538.
17. Gabriely, Y.; Rimon, E. Competitive online coverage of grid environments by a mobile robot. *Comput. Geom.* **2003**, *24*, 197–224. [\[CrossRef\]](#)
18. Dakulović, M.; Čikeš, M.; Petrović, I. Efficient Interpolated Path Planning of Mobile Robots based on Occupancy Grid Maps. *IFAC Proc. Vol.* **2012**, *45*, 349–354. [\[CrossRef\]](#)
19. Klančar, G.; Seder, M.; Blažič, S.; Škrjanc, I.; Petrović, I. Drivable Path Planning Using Hybrid Search Algorithm Based on E\* and Bernstein-Bézier Motion Primitives. *IEEE Trans. Syst. Man Cybern. Syst.* **2019**, *51*, 4868–4882. [\[CrossRef\]](#)
20. Le, A.V.; Prabakaran, V.; Sivanantham, V.; Mohan, R.E. Modified A-Star Algorithm for Efficient Coverage Path Planning in Tetris Inspired Self-Reconfigurable Robot with Integrated Laser Sensor. *Sensors* **2018**, *18*, 2585. [\[CrossRef\]](#) [\[PubMed\]](#)
21. Lui, Y.T.; Sun, R.Z.; Zhang, T.Y.; Zhang, X.N.; Li, L.; Shi, G.Q. Warehouse-Oriented Optimal Path Planning for Autonomous Mobile Fire-Fighting Robots. *Secur. Commun. Netw.* **2020**, *2020*, 6371814. [\[CrossRef\]](#)
22. LaValle, S.M. *Planning Algorithms*; Cambridge University Press: Cambridge, MA, USA, 2006; p. 1007.
23. Shrivastava, K.; Kumar, S. The Effectiveness of Parameter Tuning on Ant Colony Optimization for Solving the Travelling Salesman Problem. In Proceedings of the 8th International Conference on Communication Systems and Network Technologies, Bhopal, India, 24–26 November 2018. [\[CrossRef\]](#)
24. Shweta, K.; Singh, A. An Effect and Analysis of Parameter on Ant Colony Optimization for Solving Travelling Salesman Problem. *Int. J. Comput. Sci. Mob. Comput.* **2013**, *2*, 222–229.
25. Hopfield, J.; Tank, D. Neural Computation of Decisions in Optimization Problems. *Biol. Cybern.* **1985**, *52*, 141–152. [\[CrossRef\]](#)
26. García, L.; Talaván, P.; Yáñez, J. Improving the Hopfield model performance when applied to the traveling salesman problem. *Soft Comput.* **2017**, *21*, 3891–3905. [\[CrossRef\]](#)
27. Shi, Y.; Zhang, Y. The neural network methods for solving Traveling Salesman Problem. *Procedia Comput. Sci.* **2022**, *199*, 681–686. [\[CrossRef\]](#)
28. Gabriely, Y.; Rimon, E. Spiral-STC: An On-Line Coverage Algorithm of Grid Environments by a Mobile Robot. In Proceedings of the IEEE International Conference on Robotics and Automation, ICRA'02, Washington, DC, USA, 11–15 May 2002; Volume 1, pp. 954–960. [\[CrossRef\]](#)
29. Kan, X.; Teng, H.; Karydis, K. Online Exploration and Coverage Planning in Unknown Obstacle-Cluttered Environments. *IEEE Robot. Autom. Lett.* **2020**, *5*, 5969–5976. [\[CrossRef\]](#)
30. Hassan, M.; Liu, D. PPCPP: A Predator–Prey-Based Approach to Adaptive Coverage Path Planning. *IEEE Trans. Robot.* **2020**, *36*, 284–301. [\[CrossRef\]](#)
31. Yang, C.; Tang, Y.; Zhou, L.; Ma, X. Complete Coverage Path Planning Based on Bioinspired Neural Network and Pedestrian Location Prediction. In Proceedings of the 2018 IEEE 8th Annual International Conference on CYBER Technology in Automation, Control, and Intelligent Systems (CYBER), Tianjin, China, 19–23 July 2018; pp. 528–533. [\[CrossRef\]](#)
32. Dogru, S.; Marques, L. ECO-CPP: Energy constrained online coverage path planning. *Robot. Auton. Syst.* **2022**, *157*, 104242. [\[CrossRef\]](#)
33. Di Franco, C.; Buttazzo, G. Energy-Aware Coverage Path Planning of UAVs. In Proceedings of the 2015 IEEE International Conference on Autonomous Robot Systems and Competitions, Vila Real, Portugal, 8–10 April 2015; pp. 111–117. [\[CrossRef\]](#)
34. Savkin, A.V.; Huang, H. Asymptotically Optimal Path Planning for Ground Surveillance by a Team of UAVs. *IEEE Syst. J.* **2022**, *16*, 3446–3449. [\[CrossRef\]](#)
35. Dubin, L.E. On curves of minimal length with constraint on average curvature, and with prescribed initial and terminal positions and tangents. *Am. J. Math.* **1957**, *79*, 497–516. [\[CrossRef\]](#)
36. Backman, J.; Piirainen, P.; Oksanen, T. Smooth turning path generation for agricultural vehicles in headlands. *Biosyst. Eng.* **2015**, *139*, 76–86. [\[CrossRef\]](#)

37. Yu, X.; Roppel, T.A.; Hung, J.Y. An Optimization Approach for Planning Robotic Field Coverage. In Proceedings of the 41st Annual Conference of the IEEE Industrial Electronics Society, Yokohama, Japan, 9–12 November 2015. [\[CrossRef\]](#)
38. Jin, J.; Tang, L. Optimal Coverage Path Planning for Arable Farming on 2D Surfaces. *Trans. ASABE* **2010**, *53*, 283–295. [\[CrossRef\]](#)
39. Lee, T.K.; Baek, S.; Choi, Y.H.; Oh, S.Y. Smooth coverage path planning and control of mobile robots based on high-resolution grid map representation. *Robot. Auton. Syst.* **2011**, *59*, 801–812. [\[CrossRef\]](#)
40. Brezak, M.; Petrović, I. Real-time Approximation of Clothoids With Bounded Error for Path Planning Applications. *IEEE Trans. Robot.* **2014**, *30*, 507–515. [\[CrossRef\]](#)
41. Lepetič, M.; Klančar, G.; Škrjanc, I.; Matko, D.; Potočnik, B. Time optimal path planning considering acceleration limits. *Robot. Auton. Syst.* **2003**, *45*, 199–210. [\[CrossRef\]](#)
42. Kanayama, Y.; Kimura, Y.; Miyazaki, F.; Noguchi, T. A stable tracking control method for an autonomous mobile robot. In Proceedings of the IEEE International Conference on Robotics and Automation, Cincinnati, OH, USA, 13–18 May 1990; Volume 1, pp. 384–389. [\[CrossRef\]](#)
43. Seder, M.; Baotić, M.; Petrović, I. Receding Horizon Control for Convergent Navigation of a Differential Drive Mobile Robot. *IEEE Trans. Control. Syst. Technol.* **2017**, *25*, 653–660. [\[CrossRef\]](#)