Design of UVC Surface Disinfection Robot with Coverage Path Planning Using Map-Based Approach At-The-Edge

Sen Wang 1,2, Yongyao Li 1,2, Guanyu Ding 1, Chao Li 1, Qinglei Zhao 3,*, Bingbing Sun 4 and Qi Song 1,5,*

1 Pilot AI Company, Hangzhou 310000, China
2 School of Electronic and Information Engineering, Changchun University of Science and Technology, Changchun 130012, China
3 Changchun Institute of Optics, Fine Mechanics and Physics, Chinese Academy of Sciences, Changchun 130033, China
4 State Key Laboratory of Fine Chemicals, Dalian University of Technology, Dalian 116024, China
5 Suzhou Institute of Biomedical Engineering and Technology, Chinese Academy of Sciences, Suzhou 215163, China

* Correspondence: coldsun@ciomp.ac.cn (Q.Z.); songqi@sibet.ac.cn (Q.S.)

Abstract: In response to the issue of virus contamination in the cold-chain warehouse or hospital environment under the influence of the COVID-19, we propose the design work of a disinfection robot based on the UVC radiation mechanism using the low-computational path optimization at-the-edge. To build a surface disinfection robot with less computing power to generate a collision-free path with shorter total distance in studies, a 2D map is used as a graph-based approach to automatically generate a closed-loop disinfection path to cover all the accessible surfaces. The discrete disinfection points from the map are extracted with effective disinfection distances and sorted by a nearest-neighbor (NN) search over historical trajectory data and improved A* algorithm to obtain an efficient coverage path to all accessible boundaries of the entire area. The purpose of improved A* algorithm with NN is not to find the optimal path solution but to optimize one with reasonable computing power. The proposed algorithm enhances the path-finding efficiency by a dynamically weighted heuristic function and reduces the path turning angles, which improves the path smoothness significantly requiring less computing power. The Gazebo simulation is conducted, and the prototype disinfection robot has been built and tested in a real lab environment. Compared with the classic A* algorithm, the improved A* algorithm with NN has improved the path-finding efficiency and reduced the path length while covering the same area. Both the simulation and experimental results show that this approach can provide the design to balance the tradeoffs among the path-finding efficiency, smoothness, disinfection coverage, and computation resources.

Keywords: disinfection robot; UVC radiation; indoor disinfection; disinfection path optimization; nearest-neighbor (NN) sorting; improved A* algorithm

1. Introduction

With the rapid development of artificial intelligence and robotics technology in recent years, the use of mobile robots in people’s daily life and industrial applications has grown significantly [1–3]. The outbreak of the COVID-19 pandemic globally from early 2020 has brought significant threats to people’s lives and health [4]. Many countries across the world have reported coronavirus contamination for the cold-chain food in the warehouse environment. Hospital, shopping malls, and airports are other types of environments facing serious virus contamination. However, most scenarios employ manual viral sterilization, which is inefficient at providing enough disinfection [5]. The mobile robot can replace manual disinfection activities, avoid contact between personnel and the virus, lower the risk of spread among people, and improve disinfection efficiency. Surface contact, in addition to airborne transmission, is a mode of transmission for COVID-19 [6]. Objects or goods placed
in those environments may contain a significant amount of coronavirus on the surface. Therefore, developing a disinfection robot which can automatically and efficiently sterilize all the surfaces potentially exposed to the virus is highly desired.

UVC ultraviolet light has a wavelength range of 200–280 nm and can kill microorganisms by destroying DNA (deoxyribonucleic acid) or RNA (ribonucleic acid) in their cells. UVC irradiation has been proven to suppress live COVID-19 in studies, including on different objects requiring varying UV radiation levels [7,8]. UVC disinfection equipment emits light into the surrounding space, killing bacteria/viruses in the air or on the surfaces of objects. As a result, UVC light disinfection is an effective approach for sterilizing items in indoor scenarios, has environmental and energy-saving benefits [9–11], and plays an essential role in the management and prevention of the COVID-19 pandemic [12,13]. Our collaborative project with Bingbing Sun’s group in Dalian University of Technology conducted research on 254 nm wavelength UVC light inactivation on SARS-CoV-2 to break the transmission. The relationship of UVC radiation dose with inactivation rate has been established from a scientific experiment by a qualified third-party testing agency. The result shows that in the range of 16 cm with a nominal radiation intensity of 14 mJ/cm², 99.99% of the SARS-CoV-2 virus could be inactivated and the transmission chain could be broken [14]. This gave us the idea that COVID-19 virus could be precisely inactivated by managing the distance between UVC light (disinfection robot) and the contaminated surface. Figure 1a,b present the microscope views of MHV-A59 coronavirus samples without UVC radiation and with a UVC dose of 14.0 mJ/cm², respectively. The virus titers in Figure 1a are $10^{9.625}/\text{mL}$. Results indicates that 99.99% of the virus could be inactivated with enough UVC dose. Furthermore, the empirical relationship of the distance from the robot to the target surface and disinfection time has been used to guide the robot for an effective disinfection in Figure 1c. UVC radiation dose could be calculated by multiplying radiation intensity and time, as in Equation (1). The distance and radiation could be described as a polynomial-fitting curve in Equation (2) to fit the data, where $x$ is the distance in centimeter, and $W$ is the radiation intensity in mW/cm². We noted that some reports may use quite complicated equations to describe the relationship, while we use simple math to give an acceptable result in a straightforward way. By understanding this principle, we can control the disinfection time properly with respect to distance, which corresponds to the UVC radiation dose constraint to inactivate the COVID virus on surfaces.

\begin{equation}
E = W \times t
\end{equation}

\begin{equation}
W = \frac{40}{x^{1.2}}
\end{equation}

As illustrated in Figure 2, the UVC disinfection robot in this work consists of an intelligent control unit, a UVC disinfection unit, and the scheduling unit. To plan the robot’s running trajectory and manage the disinfection work process, the intelligent control unit measures the distance information around the robot. The UVC disinfection unit radiates the object surfaces after receiving the execution instructions and disinfection mode from the intelligent control module. The scheduling unit communicates with the UVC disinfection unit via cloud protocol and performs remote control. Therefore, the robot has the capability of adjusting UVC irradiation time based on the distance information.

Coverage Path Planning (CPP) is the task of finding a path that passes over all points of an area or places of interest while avoiding obstacles. Different from classic CPP, the goal of disinfection path-planning is for every point on an object surface to be visible on the robot trajectory, and every point in an object surface must receive a certain amount of irradiance exposure. This scenario induces a joint problem of robot trajectory planning and disinfection time assignment. Since the problem is a complex non-convex optimization problem and NP-hard, there is some interesting work to explore the optimized coverage path. Feng has reported the sensor coverage optimization targeting 2D surfaces embedded in 3D workspace [15]. Marques proposed a two-stage scheme to approximate the solution of the induced NP-hard optimization for irradiance efficiency, and results show that it can
achieve more coverage for the same exposure time as strategies for existing UV robots, although this method needs a high-performance GPU unit to calculate radiation dose and is difficult to deploy for a practical environment [16]. In this work, we propose an intuitive way to generate optimal trajectories from 2D SLAM maps to cover all accessible points. All points are extracted from a 2D map contour with “effective distance D” to keep UVC dosing constraints, and the sequence of disinfection points are treated as a traveling salesman problem (TSP). Figure 3 shows a diagram to illustrate how the radiation power is estimated while the robot is moving at the speed of V. With effective distance D to the disinfection point A, the UVC radiation power can be obtained as

\[
E = \int_{l_1}^{l_3} W(l) \times t
\]  

(3)

where radiation power \(E\) could be obtained as the integral of radiation intensity \(W(l)\) with travel time \(t\), and \(l\) is the distance from robot to disinfection point. Here, we define the effective time zone that the robot is traveling with the constant speed \(V\) from the moment...
from $t_1$ to $t_3$. $W(l)$ can be determined from fitting the curve in Figure 1c. To meet the UVC dose constraint of 14.0 mJ/cm$^2$, the effective distance $D$ can be determined from the geometry rule in Figure 3. Bear in mind that $D$ will be used to generate disinfection point clusters away from the map contour later in this work. Thus far, by setting the effective disinfection distance to all surfaces, we have turned the UVC radiation dose-control problem into a path-planning problem.

![Diagram of UVC radiation power estimation.](image)

**Figure 3.** Diagram of UVC radiation power estimation.

The A* algorithm is usually used to improve the path-planning system based on the heuristic search. The heuristic function is used to lead the node search towards the target node, which improves the efficiency of finding the ideal path and has high portability and adaptability [17–21]. The classic A* algorithm, on the other hand, has some drawbacks, as expected. The path-finding efficiency is relatively low due to the node search problem, and the planned path contains numerous intricate turning points with low smoothness. As a result, several researchers have conducted research on the improved A* algorithm. Bo Zhang et al. proposed a hierarchical path-planning approach for a wheeled robot to move in a partially known uneven terrain. This method combined the classical A* algorithm and Q-learning method, which offered pleasing approaches that avoid locally recognized impediments and impassable areas in a partially known uneven terrain [22]. The A* algorithm was improved in two ways by B. Fu and L. Chen et al. One option is to plan a local path between the current node and the target node in the current node’s search area if not colliding. The first attempt is to use the local path; the second is to apply optimization to the estimated path in order to shorten it and improve its smoothness [23]. M. Lin and K. Yuan et al. improved the real-time performance of the A* algorithm by including the parent node of the current node in the heuristic function, implementing a heuristic function that matches the optimal weight, and adding the parent node of the current node to the heuristic function [24]. H.M. Zhang and M.L. Li et al. improved the A* algorithm’s evaluation function while maintaining a reasonable level of security. A path evaluation method is also employed to smooth the path, significantly improving the robot’s movement safety in the complicated terrain as well as the path’s smoothness [25–27]. This research addresses the drawbacks of the classic A* algorithm from two aspects. First, to increase the path-finding efficiency, a dynamically weighted heuristic function is utilized. Second, to improve the efficiency of the global search and the path smoothness, the path turning angles have been reduced by screening on redundant turning points.

On the basis of the UVC disinfection mechanism and novel path-planning from the map presentation, the design of the UVC disinfection robot using the low-computational path optimization has been proposed and demonstrated. The closed-loop path is generated to cover all the accessible surfaces in the map. In particular, the discrete disinfection points are extracted from the map with effective disinfection distance. The points are sorted by the nearest-neighbor (NN) search and improved A* algorithm to obtain an efficient coverage path to reach out all surfaces. The purpose of the improved A* algorithm with NN used here is not to find the optimal solution but to optimize one with low computing power. Both simulation and experimental results have demonstrated that the proposed algorithm improves the path-finding efficiency and path smoothness significantly by reducing the path turning angles with low power. The paper shows the full design workflow of disinfection robot which can be applied to a low-cost robot platform.
2. The Proposed Method

In this section, the proposed method is elaborated as follows. First, the map needs to be obtained in the presence of the binary image. Then, a point-generation algorithm is applied to extract a group of the discrete disinfection points from the contour of the objects, such as shelves, checkboxes, chairs, and boundaries, from the map. Meanwhile, the discrete disinfection points are associated with the defined disinfection distance along the vertical direction according to the object’s surface. The disinfection spots are sorted using the nearest-neighbor (NN) sorting algorithm to provide a list of disinfection points consequentially. Finally, by traversing the sorted disinfection points and using the improved A* algorithm, the disinfection point sequence is taken as sub-starting points and sub-target points to form a closed-loop sterilization path to cover all possible boundaries.

2.1. Discrete Disinfection Points Generation Algorithm

Creating a reasonable amount of distinct disinfection points from the map is the first step toward achieving an effective global disinfection path. By calling the discrete disinfection points generation algorithm, the object’s contour is extracted and fitted back to the map image, and the disinfection points that meet the distance requirements are generated in the effective vertical direction of the object’s surface according to the contour direction for the subsequent global disinfection path planning.

(1) Map image preprocessing: OpenCV library is utilized to preprocess the map image in order to accurately extract the contours of the disinfection regions and objects in the map and then generate efficient disinfection spots that fulfill the distance requirements along the vertical direction of the surface.

By performing SLAM mapping on the disinfection area, the occupation grid map has been created [28,29]. In such a map, a pixel is equal to each grid in the grid map. The coordinate position of the spot where the laser beam impacts is determined during the lidar scanning procedure. Where there is reflection, the pixel value corresponds to a gray value; where there is no reflection, the pixel corresponds to a non-gray value. At the same time, each grid has just two states, implying that the grid’s position may not be occupied by the objects at all or may be illuminated multiple times. When a grid is illuminated several times, it implies that the grid has been occupied, which means there is an obstruction in the grid’s path. As a result, the occupancy grid map uses the image’s gray value to represent the chance of occupancy and the occupancy probability to express the possibility that the grid’s location is an obstacle.

The morphological modification of the occupied raster map image is acquired by first executing an erosion operation and then expanding the image to obtain a grayscale image, which effectively removes some small, closed sections in the image. The picture magnification is configured in the algorithm for various sized maps in order to set disinfection points properly. A global threshold can be specified, and the original map picture can be scaled according to the magnification. To achieve binarization of the image, the threshold is set to 220 to create the gray (unknown) and black areas (obstacle), which are considered inaccessible areas; the accessible area gray value is set to 255, while the inaccessible area gray value is set to 0. To reliably generate accessible disinfection spots, the object edge is searched first on the binary picture, then the contour detection of the obstacle is conducted. Polygon fitting is applied to the contour points obtained in the previous phase, as well as to the contour points of the image, using polyline continuous smooth curves, and a series of fitted contour points is processed using polygon approximation.

(2) The principle of generating discrete disinfection points: The image, after edge detection and a sequence of object outline points after polygon fitting, can be derived from the previous section’s preprocessing of the map image. Sterilization point generation is based on connecting two nearby object points on a map. In the vertical direction of the line, disinfection points that meet the distance requirements are generated. Because the object edge’s line on the map includes two vertical directions, it is necessary to first determine the effective vertical direction between the object contour points before generating effective
disinfection spots that fulfill the accessible area’s distance and angle requirements. The disinfection distance threshold is specified in the algorithm to ensure that the disinfection robot can radiate the surface sufficiently and keeps a safe distance at the same time. As a result, the distance between the disinfection points and the items equals the sum of the disinfection distance threshold and the robot’s radius; furthermore, if the disinfection distance threshold is set larger, the disinfection time of the disinfection robot will be longer so as to ensure effective disinfection on the surfaces. The algorithm also establishes a threshold for the spacing between disinfection points in order to adapt to varied maps and create disinfection points flexibly according to actual disinfection tasks. Disinfection points can be formed at the distance along the edge of the objects adjacent to the objects. Depending on the requirements of actual disinfection duties, the disinfection points generated in the map could be denser or scattered when the interval threshold is set to be smaller or larger. In order to make the coverage high with high search efficiency of the disinfection path-planning, it is necessary to accurately set the disinfection point interval threshold according to the actual disinfection task and map. Assuming that the boundaries in a house are the objects, the disinfection points generated by the disinfection point generation algorithm in the map is shown as the red line in Figure 4.

![Figure 4. Disinfection points generated around the objects in the map.](image)

We find the set of adjacent fitted contour points \( p = \{ (x_j, y_j) | j = 1, 2, \ldots, n \} \) whose distance is larger than the threshold of the distance between disinfection points by traversing the set of fitted contour points for obstructions (assuming that there are \( n \) points in the set of adjacent contour points that meet the distance requirements). In the vertical direction of the line linking nearby contour points, disinfection spots that match the distance requirements are created using geometric principles. Figure 5 depicts the generation concept.

Assuming that the first two fitted contour points in the set \( p \) are \( a \) and \( b \), the point \( c \) is placed on the connecting line between \( a \) and \( b \) at a certain distance from point \( a \), a normal line of line \( ab \) is drawn through point \( c \), and the distance from point \( c \) is the disinfection distance threshold point \( d \). \( d \) is the first initial disinfection point. The pixel coordinates of point \( a \) are \((x_a, y_a)\), and the pixel coordinates of point \( b \) are \((x_b, y_b)\), the distance between...
points \( a \) and \( c \) is \( m \), the disinfection distance threshold is \( L \), where \( dc = L \), and the geometric principle can be obtained as:

\[
\sin \alpha = -\frac{y_b - y_a}{\sqrt{(x_b - x_a)^2 + (y_b - y_a)^2}}, \quad \cos \alpha = \frac{x_b - x_a}{\sqrt{(x_b - x_a)^2 + (y_b - y_a)^2}}
\]

Thus, the pixel coordinates \((x_d, y_d)\) of the first disinfection point \( d \) are:

\[
\begin{align*}
    x_d &= x_a + m \cos \alpha + L \sin \alpha \\
    y_d &= y_a - m \sin \alpha - L \cos \alpha
\end{align*}
\]

The orientation angle of the line \( cd \) is: \( \beta = 90^\circ - \alpha \).

---

**Figure 5.** Schematic diagram of disinfection points generation.

The steps to generate discrete disinfection points are as follows:

1. Perform preprocessing operations such as morphological changes, image binarization, image edge detection, and contour detection on the map image and extract the shelf edge contour point set \( p = \{(x_j, y_j)\} \) after polygon fitting.

2. Detect and filter out whether the distance between adjacent contour points in the contour point set \( p = \{(x_j, y_j)\} \) is greater than the distance threshold between the disinfection points.

3. On the basis of the outline of the object in the map, determine the effective vertical direction between the lines connecting adjacent contour points, generate disinfection points in the effective vertical direction between the lines connecting adjacent contour points, and generate along the contour direction based on the distance threshold between the disinfection points.

4. Verify whether the disinfection points created in the previous step match the distance and angle requirements.

(3) Nearest-neighbor (NN) sorting of disinfection points: To avoid producing too many paths by forming a closed path based on discrete disinfection sites, the disinfection robot’s disinfection effectiveness must be improved by sorting the created initial disinfection points using the closest-neighbor sorting method.

To perform the disinfection task, the disinfection points are filtered out for the closest point to the origin of the pixel coordinate system in the map from the disinfection points listed as the first disinfection point \( p_1 \). Record this disinfection point as the current disinfection point \( p_R \) and delete \( p_1 \) from the disinfection point list; then, traverse the disinfection point list to find the disinfection point closest to the current disinfection point from the remaining disinfection points as the second disinfection point \( p_2 \) to perform the disinfection task. Record \( p_2 \) as the current disinfection point \( p_R \) and remove \( p_2 \) from the disinfection point list, and so on. The NN disinfection point list is generated after all of the disinfection
points in the disinfection point list have been sorted. Figure 6 depicts the disinfection point NN sorting procedures.

Figure 6. Flowchart of NN sorting for disinfection points.

2.2. A Global Disinfection Path-Planning Algorithm Based on Improved A *

The pixel coordinate system of the map image is used to plan the global path in this work. The disinfection point global path-planning algorithm obtains the global path’s pixel coordinates, which are then transformed into two-dimensional physical coordinates under the map so that the disinfection robot can execute it. As a result, before doing global path-planning, the original map image should be converted to a grayscale image, which should then be converted to a binary image with only grayscale values of 255 and 0. The pixels with a grayscale value of 255 are among them. The navigable area is represented by the area, while the obstacle area is represented by the pixel area with the gray value of 0. Then, the two-dimensional physical coordinates of the disinfection points in the NN disinfection point list are converted into pixel coordinates, and the improved A * algorithm treats these disinfection points as sub-starting points and target points for path-planning to form a closed global disinfection path. The disinfection path information is stored in the global disinfection path executor function package.

The improved A * algorithm can make an optimal path from the starting point to the end point. The first disinfection point in the NN disinfection point list is selected as the first sub-starting point of the improved A * path-planning. The second disinfection point in the point list is used as the first sub-goal point of the improved A * path-planning. When the path-finding is successful, the first sub-target point is set as the second sub-starting point, and the third disinfection point in the NN disinfection point list is used as the second target
point for improving A* path-planning, and so on. Until the path-planning between all disinfection points in the NN disinfection point list is completed, a closed-loop disinfection path from the first disinfection point to the last disinfection point in the list is formed.

(1) Conversion between ROS map coordinate system and pixel coordinate system: As shown in Figure 7, the origin \(O_p\) of the pixel coordinate system is in the upper left corner, while the origin \(O_r\) in the map coordinate system in the ROS system is in the lower left corner; therefore, the conversion relationship between the pixel coordinate \(d_p(x_p, y_p)\) of the disinfection point and the coordinate \(d_r(x_R, y_R)\) of the disinfection point in the ROS map is:

\[
\begin{align*}
\begin{cases}
x_p = x_R - \frac{ω_x}{m_R} - (\frac{y_n}{m_R} - \frac{ω_y}{m_R}) \\
x_p = H - (\frac{y_n}{m_R} - \frac{ω_y}{m_R})
\end{cases}
\]

where \(m_R\) is the resolution of the map in ROS, \((ω_x, ω_y)\) the initial position coordinates of the robot when building the map, and \(H\) is the number of rows of pixels in the map image.

![Figure 7. Pixel coordinate system and ROS map coordinate system.](image)

(2) Optimization evaluation function of the improved A* algorithm: Although the classic A* algorithm can find a shorter path, due to its evaluation function, its path search efficiency has to be improved, as there are many redundant turning points in the planned path, resulting in a large path turning angle and low path smoothness. This research improves the classic A* algorithm by focusing on these two issues, and the flowchart of the improved A* algorithm is shown in Figure 8.

The evaluation function \(h_0(n)\) of the classic A* algorithm is shown in Formula (6), where \(n\) is the current node in the path planning process, and \(g\) is the target node, so that \((x_n, y_n)\) are the coordinates of the current node in the grid map, and \((x_g, y_g)\) are the coordinates of the target point in the grid map. The heuristic function \(h_0(n)\) plays a guiding role in the evaluation function to search for nodes in the direction of the target point for path-planning.

\[
h_0(n) = \sqrt{(x_n - x_g)^2 + (y_n - y_g)^2}
\]

The search efficiency and flexibility can be improved by modifying the heuristic function. When the path search is in the initial stages, there are more optional nodes such that a quick search towards the target point will clearly be preferable. When the node is close to the vicinity of the target point, obtaining the shortest path toward the target direction will dominate. Therefore, a dynamic-weighted heuristic function is proposed to optimize the evaluation function, which will set up the weight coefficient and the size of the coefficient of \(h_0(n)\). The distance threshold is set as below:

\[
H = \sqrt{\frac{(x_0 - x_g)^2 + (y_0 - y_g)^2}{C}}
\]

where \((x_0, y_0)\) is the starting point coordinate, \(C\) is a constant greater than zero, and \(H > c\). When the distance to the target point is greater than the threshold, the weight of the heuristic function can be increased to improve the search efficiency of the algorithm. In other words, the coefficient of \(h_0(n)\) should increase with \(h_0(n)\). At this time, the algorithm will be guided to search in the direction of the target point to improve the search efficiency.
When the distance to the target point is less than the threshold, such as in the case when
the path search is approaching the target point, $h_0(n)$ and the coefficient of $h_0(n)$ are small
and the heuristic function should remain unchanged to improve the cost function. The
proportion of $g(n)$ in the evaluation function represents the potential of finding the shortest
path. The optimized heuristic function can be written as:

$$m(n) = \begin{cases} h_0(n), & (h_0(n) < H) \\ ln(h_0(n)) \cdot h_0(n), & (h_0(n) \geq H) \end{cases}$$

(8)

Figure 8. Flow chart of the improved A * algorithm.

The improved evaluation function is thus:

$$f(n) = g(n) + m(n)$$

(9)

where $g(n)$ is the cost from the current node to the starting point.

(3) Path-smoothing strategy of the improved A * algorithm: Because the turning angle
of the path obtained by the classic A * algorithm is randomly generated and sometimes too
large or not smooth enough, which is undesired for the robot, a path-smoothing approach
is given in this work to eliminate redundant turning spots. By traversing the path nodes
in the grid map, the starting node of a sub-path is set as $a(x_1, y_1)$, the intermediate node
as $b(x_2, y_2)$, and the successor node of $b$ as $c(x_3, y_3)$. If three continuous path nodes are
linear, Formula (7) is met and the linear nodes are reserved. If not, $b$ is regarded as a
redundant turning point. If there is no obstacle between $a$ to $c$, node $b$ is deleted and the
smooth path is $a \rightarrow c$. Through this strategy, all original waypoints are processed to obtain
a smoother path.

$$(y_3 - y_1)(x_2 - x_1) - (y_2 - y_1)(x_3 - x_1) = 0$$

(10)
2.3. Disinfection Path Executor

(1) The mechanism of the disinfection path executor: The move_base package offers the configuration, operation, and interaction interface for ROS navigation to fulfill the robot navigation task. The move_base primarily consists of the global and local planners associated with the cost_map, respectively. The navigation task will invoke the global planner and the local planner to execute global path-planning and local obstacle avoidance. The actionlib function package in ROS can implement the communication with a feedback mechanism. Action is a type of Request-and-Answer communication mechanism similar to Service. The difference is that Action has a feedback mechanism, which can continually feedback on the task progress in “real-time” and can abort the operation in the runtime.

The disinfection path executor and the move_base package communicate through the Action communication mechanism to control the disinfection robot to execute the global disinfection path. In essence, the move_base package is set to be the Action server in communication, and the disinfection path executor is set as the Action client. The communication between the disinfection path executor and move_base is shown in Figure 9.

![Figure 9. The relationship between the disinfection path executor and move_base.](image)

The code structure of the disinfection path executor is a basic state machine. For this reason, it is easy to add complex behaviors controlled by defined transitions in the Python scripts. State transitions in the program occur in the order GET_PATH, FOLLOW_PATH, and PATH_COMPLETE. The disinfection path executor uses the global disinfection path point generated by the disinfection path planning algorithm to send the move_base_msgs/MoveBaseAction in the form of a series of target points using actionlib data to the move_base server. After receiving this action, the disinfection robot starts to execute each disinfection path point in the sequence, and continuously feedbacks on the current movement state during the execution. If the robot successfully reaches the global disinfection waypoints, the Action returns a “task completion” flag.

By publishing the waypoints topic, the disinfection path executor publishes the waypoint data in the current queue in the geometry_msgs/PoseArray data format, and this waypoints information can be displayed in Rviz in real-time. At the same time, the disinfection path executor subscribes to the std_msgs/Empty message in the path-ready topic to initialize the path point, and after executing a path point in the queue, clears the waypoint sequence by subscribing to the std_msgs/Empty message in the path_reset topic.

(2) Path-smoothing strategy for disinfection path executor: The disinfection path executor aids in the work of handling several waypoints and publishes log information output, relying on the fundamental navigation scheme. The disinfection path executor will automatically publish each path point’s coordinates and adjust the acceleration/deceleration...
distance as needed. The disinfection path executor uses a path-smoothing approach to lessen the robot’s acceleration or deceleration during path point execution and lets the disinfection robot walk more smoothly along the disinfection path. If the current execution path point sequence is a straight path or the disinfection robot is close enough to the current path target point, the next path target point will be released in advance to replace the current path target point. Thus, the disinfection robot will not experience the deceleration for the current target point, ensuring the smooth transition if, along the curve path, the path adjustment strategy is not to release the target point in advance. In other words, when the disinfection robot reaches the current target point, it will decelerate to 0 and then go to the next target point to achieve the maximum degree of path fit. At the same time, the disinfection path executor will publish the path execution log, which will record the disinfection path point’s execution status, such as whether the path was reached, as well as the path’s fit. To behave more efficiently, if the robot fails to reach the goal point ten times in a row or if the program decides that the robot is stuck, the disinfection path executor will terminate the execution automatically.

In an example of how the smooth threshold works in the disinfection path executor, three path points A, B, and C are chosen to be in a straight line. The disinfection path executor issues a task to C when the disinfection robot arrives at A and is within a certain distance to B. When the smoothing threshold is 0, the disinfection robot will decelerate in advance before reaching point B, stop at point B, and then release the task to point C. The larger the threshold, the better it can avoid the problem of deceleration. However, if it is too large, it may cause a path-not-fit issue, which is undesirable for robot control.

3. Experimental Results

3.1. Simulation Experiment of Global Path Planning Algorithm Based on Improved A *

A simulation is conducted in the Python program to compare the performance of the improved A * algorithm and the classic A * algorithm. In particular, the open list of nodes has been created and searched. For each node x in the open list, the distance from the start is called g(x). The open list starts out containing the start node with distance 0 from itself. For each search step, the node x with the lowest value of g(x) + h(x) is selected, which is estimated to be closest to a goal. Here, g(x) is the distance finished, h(x) is the distance estimated to go, and the sum is the estimated total distance. If x is a goal, it is finished and moves to the next goal. Otherwise, x is removed from the open list, and the neighbors which have not already been visited are inserted.

In the simulation, a 50 × 50 grid map has been defined on the experimental platform, and random obstacles are inserted in the map in which each grid is set as a 1 cm-long square. The radius of the disinfection robot was 0.4 cm, and the safety distance was set as 0.6 cm. Figures 10 and 11 show the simulation results of the classic A * algorithm and the improved A * algorithm in the map, where the purple point represents the disinfection robot’s starting point, and the blue point represents the target point; the black regions are the obstacles, and the red area defines the collision area. The green line shows the simulated global path. Table 1 shows the comparison of two algorithms’ performances.

<table>
<thead>
<tr>
<th>Algorithm Category</th>
<th>Path Length (cm)</th>
<th>Path Search Time (s)</th>
<th>Path Turning Angle (Deg)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Classic A * algorithm</td>
<td>79.84</td>
<td>39</td>
<td>855</td>
</tr>
<tr>
<td>Improved A * algorithm</td>
<td>77.96</td>
<td>15</td>
<td>397</td>
</tr>
</tbody>
</table>

According to Figures 10 and 11 and Table 1, the improved A * algorithm improves the path search efficiency by 61.5%, compared with the classic A * algorithm. In terms of path length, the improved A * algorithm reduces by 2.4%, compared with the classic
A * algorithm. In terms of path turning angle, the improved A * algorithm reduces 53.6%, compared with the classic A * algorithm. Due to the limitation of the heuristic function evaluation function, the classic A * algorithm in the search direction of the lack of certain guidance in search efficiency is low; the improved A * algorithm applies the distance threshold and dynamically weighted heuristic function according to the distance threshold, which significantly improves the efficiency of node searching. As a result, it outperforms the classic A * algorithm in algorithm execution efficiency. The path-smoothing approach is then employed to reduce the turning angle as well as the path length, thus improving the global path planning effect.

![Figure 10](image1.png)

**Figure 10.** The path obtained by the classic A * algorithm with the 50 × 50 grid map.

![Figure 11](image2.png)

**Figure 11.** The path obtained by the improved A * algorithm with the 50 × 50 grid map.

3.2. Simulation Experiment of Global Disinfection Path in ROS

In this experiment, Gazebo physical engine in ROS is utilized to mimic the behavior of a two-differential chassis; in order to verify the feasibility of the discrete disinfection points generation algorithm, the disinfection path-planning is based on the improved A * and the
disinfection path executor. A popular jackal is used to create a closed area to be disinfected. URDF models of the disinfecting robot is configured properly and added to Gazebo for all physical sensors and tf relation. The simulation environment model established in classic jackal map is shown in Figure 12, which is 227.5 square meters with a resolution of 0.05 m. The simulation experiment was carried out in Ubuntu16.04 with ROS Kinetic. The processor of the computer was i7-6700, the main frequency was 3.4 GHZ, and the operating memory was 4 GB.

Figure 12. The simulation jackal model in Gazebo.

Assume that all black borders in the enclosed area map in Figure 13 are objects to be sterilized. Firstly, the discrete disinfection points generation algorithm is used to generate the required disinfection points in the map image corresponding to the simulation environment model. Considering the size of the disinfection robot in the simulation model and the actual disinfection effect, the disinfection distance threshold set in this simulation experiment is 0.3 m, the threshold of the disinfection point distance between adjacent points is 0.5 m, and the discrete disinfection points generated in the simulation map are shown in the red line segments in Figure 13.

Because the NN order is carried out on the disinfection points before the global disinfection path is generated, the possibility of a random path without a good fit to the objects is greatly reduced. The global disinfection paths calculated by three algorithms are shown in Figure 14. Through comparison, it is clearly shown that if the NN ordering of disinfection points is not applied, there will be many invalid disinfection paths that do not fit objects well in the global disinfection path. The simulated robot has executed the global disinfection path planned by three different algorithms respectively, as shown in Figure 15. Table 2 shows the execution efficiency of the three algorithms as well as the total path length. It can be shown that if the NN sorting is not performed, a significant increase in path search time will result and the path length will also be greatly increased. The path search efficiency of global path planning of improved A * algorithm with the NN sorting can be improved to 47.6% higher than the classic A * algorithm with NN sorting, and 89.4% higher than the classic A * algorithm without NN sorting. The experimental results show that the disinfection path calculated by the improved A * algorithm with the NN sorting and the classic A * algorithm with the NN sorting can achieve almost the same disinfection
effect. In addition, due to the path-smoothing of the improved A * algorithm, the length of the disinfection path is also reduced, which is 6% less than that of the classic A * algorithm with the NN sorting and 46.3% less than that of the classic A * algorithm only.

![Figure 13. The probabilistic map of discrete disinfection points.](image)

**Figure 13.** The probabilistic map of discrete disinfection points.

![Figure 14. Global disinfection paths planned by three algorithms.](image)

(a) Improved A * algorithm with NN sorting of disinfection points; (b) Classic A * algorithm with NN sorting of disinfection points; (c) Classic A * algorithm.

**Figure 14.** Global disinfection paths planned by three algorithms. (a) Improved A * algorithm with NN sorting of disinfection points; (b) Classic A * algorithm with NN sorting of disinfection points; (c) Classic A * algorithm.

![Figure 15. The trajectories of the simulated robot executed three global disinfection paths.](image)

(a) Improved A * algorithm with NN sorting of disinfection points; (b) Classic A * algorithm with NN sorting of disinfection points; (c) Classic A * algorithm.

**Figure 15.** The trajectories of the simulated robot executed three global disinfection paths. (a) Improved A * algorithm with NN sorting of disinfection points; (b) Classic A * algorithm with NN sorting of disinfection points; (c) Classic A * algorithm.
Table 2. Path search time and path length of three algorithms in Gazebo simulation.

<table>
<thead>
<tr>
<th>Algorithm Category</th>
<th>Path Search Time (s)</th>
<th>Path Length (m)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Improved A * algorithm with NN sorting of disinfection points</td>
<td>2.77</td>
<td>165.48</td>
</tr>
<tr>
<td>Classic A * algorithm with NN sorting of disinfection points</td>
<td>5.29</td>
<td>176.06</td>
</tr>
<tr>
<td>Classic A * algorithm without NN sorting of disinfection points</td>
<td>26.19</td>
<td>308.24</td>
</tr>
</tbody>
</table>

3.3. Global Disinfection Path Experiment

The experiment was carried out in an enclosed laboratory with a length of 9.84 m and a width of 6.57 m with a map resolution of 0.05 m. The prototype of intelligent disinfection robot shown in Figure 16 has been built to verify the disinfection path optimization algorithm. A platform is mounted on a two-wheel differential chassis with a 120 W battery. Two-dimensional LiDAR and 3 × ultrasound sensors are used for robot localization and obstacle avoidance. A UVC light is attached to the platform with disinfection control module, and a status light indicates the IDLE/WORK status of the robot. Assuming that the disinfection robot needs to perform automated-cruise disinfection tasks along all the possible surfaces of all objects in the laboratory, such as the test bench, chairs, tables, and closets, the disinfection robot first uses LiDAR to generate a 2D map in the laboratory. The experimental laboratory and probability map containing the discrete disinfection points are shown in Figure 17a,b; the disinfection distance threshold of these disinfection points is 0.2 m, and the disinfection point interval threshold of adjacent points is 0.3 m.

![Figure 16. The prototype of intelligent disinfection robot.](image)

Through the guidance of discrete disinfection points, the disinfection path planning algorithm calculates an enclosed cruise disinfection path around all the boundaries of the obstacles in the laboratory. The experiment is designed in three parts—the disinfection robot performs the cruise disinfection task according to the global disinfection path calculated by the improved A * algorithm with the NN sorting, the classic A * algorithm with the NN sorting, and the classic A * algorithm only. The three global disinfection paths calculated by the three algorithms and the walking traces recorded during the execution of the three global disinfection paths are shown in Figures 18 and 19, respectively. The path-finding
efficiency and the path length calculated by the three algorithms are summarized in the following table.

![Experimental laboratory and probability map of discrete disinfection points](image1.png)

**Figure 17.** Experimental laboratory and probability map of discrete disinfection points. (a) Experimental laboratory; (b) Probability map with discrete disinfection points.

![Global disinfection paths planned by three algorithms](image2.png)

**Figure 18.** Global disinfection paths planned by three algorithms. (a) Improved A* algorithm with NN sorting of disinfection points; (b) Classic A* algorithm with NN sorting of disinfection points; (c) Classic A* algorithm only.

![The trajectories of the disinfection robot executed three global disinfection paths](image3.png)

**Figure 19.** The trajectories of the disinfection robot executed three global disinfection paths. (a) Improved A* algorithm with NN sorting of disinfection points; (b) Classic A* algorithm with NN sorting of disinfection points; (c) Classic A* algorithm only.

From the calculated disinfection paths and the walking trajectories, if the NN sorting is not applied, some invalid paths will not fit the obstacle boundaries well. Moreover, due to the large distance between non-adjacent disinfection points, the paths between these disinfection points will be close to the obstacle boundaries due to the lack of guidance of disinfection points. Such paths do not meet the requirements of disinfection distance, which will keep the robot rotating in place and possibly fail. The global disinfection paths calculated after the NN sorting are guided by adjacent disinfection points, so the paths basically fit the obstacle boundaries and meet the requirements of disinfection distance. As shown in Figure 19a,b, the actual walking trajectories of the disinfection robot are...
basically consistent with the generated disinfection path. It can be seen from Table 3 that if the NN sorting is not applied, path planning between adjacent points with reasonable distance cannot be obtained. This leads the path searching time to increase dramatically, and the corresponding path length will greatly increase as well. The search efficiency of the improved A * algorithm with the NN sorting can be boosted obviously, which is 13.6% higher than the classic A * algorithm with the NN sorting, and 75.6% higher than the classic A * algorithm only. The experimental results show that the disinfection path calculated by the improved A * algorithm with the NN sorting and the classic A * algorithm with the NN sorting can achieve almost the same performance. In addition, due to the path smoothing of the improved A * algorithm, the total length of the disinfection path is reduced as 10.7% less than that of the classic A * algorithm with NN sorting, and 38.8% less than that of the classic A * algorithm only.

Table 3. Path search time and path length of three algorithms in the lab environment.

<table>
<thead>
<tr>
<th>Algorithm Category</th>
<th>Path Search Time (s)</th>
<th>Path Length (m)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Improved A * algorithm with NN sorting of disinfection points</td>
<td>0.038</td>
<td>36.45</td>
</tr>
<tr>
<td>Classic A * algorithm with NN sorting of disinfection points</td>
<td>0.044</td>
<td>40.80</td>
</tr>
<tr>
<td>Classic A * algorithm only</td>
<td>0.156</td>
<td>59.60</td>
</tr>
</tbody>
</table>

4. Conclusions

This work presents a cost-effective design of UVC disinfection robot based on the experimental UVC radiation dose constraint with disinfection distance. A map-based coverage path-planning is designed for real-time processing at-the-edge and has been implemented with discrete point extraction and the improved A * algorithm with NN. In particular, the discrete disinfection points around all possible surfaces are generated by a discrete-point-generation algorithm from the map. The NN sorting algorithm is used to sort the disinfection points by distance to generate a disinfection points sequence. The improved A * algorithm considerably improves the searching efficiency of disinfection path by 89.4% and shortens the disinfection path length by 46.3%, compared with classic A * algorithm results in the simulation. The prototype robot system has been tested in a real lab environment. The disinfection path planning algorithm is implemented in ROS environment, and the move_base package is used to integrate the algorithm with robot hardware as the disinfection path executor. Experimental results have shown that the path-searching efficiency has been improved by 75.6% and path length been reduced by 38.8% compared with classic A * algorithm results.

It should be pointed out that this implementation is not designed to find the optimal path solution but to find an efficient CPP approach to operate in the real-time fashion with less computing resources. It can be foreseen that this algorithm could also be deployed in many robot applications, such as security surveillance robot, agricultural harvesting robot, and autonomous painting robot. Additionally, there are still some areas that need to be improved in the future. For example, the algorithm cannot identify new objects added to the environment. In the future, we will add more features such as vision-based object identification to improve the degree of automation.

Author Contributions: Conceptualization, G.D. and Q.S.; methodology, S.W. and Y.L.; software, S.W. and Y.L.; validation, S.W. and Y.L.; formal analysis, G.D.; investigation, Q.S.; resources, Q.S. and B.S.; data curation, S.W. and Q.S.; writing—original draft preparation, S.W.; writing—review and editing, Q.S., G.D. and Q.Z.; visualization, Q.S.; supervision, Q.S., C.L. and B.S.; project administration, Q.S.; funding acquisition, Q.S. and B.S. All authors have read and agreed to the published version of the manuscript.
Acknowledgments: In this section, authors acknowledge the financial support from “Feng Lu Ying Cai” incubation program and Jilin Scientific and Technological Development Program.

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

References

17. Song, Q.; Zhao, Q.; Wang, S.; Liu, Q.; Chen, X. Dynamic path planning for unmanned vehicles based on fuzzy logic and improved ant colony optimization. IEEE Access 2020, 8, 62107–62115. [CrossRef]


