



Article

An Algorithm for Painting Large Objects Based on a Nine-Axis UR5 Robotic Manipulator

Jun Wang ^{1,*} , Mingquan Yang ¹ , Fei Liang ¹, Kangrui Feng ¹, Kai Zhang ¹ and Quan Wang ²

¹ Institute of Additive Manufacturing and Industrial Robotics, School of Mechanical Engineering, Hubei University of Technology, Wuhan 430068, China; mingquan_yang@outlook.com (M.Y.); 102010139@hbut.edu.cn (F.L.); 102010085@hbut.edu.cn (K.F.); 102000015@hbut.edu.cn (K.Z.)

² Institute of Marine Special Equipment and Technology, School of Mechanical Engineering, Hubei University of Technology, Wuhan 430068, China; quan_wang2003@163.com

* Correspondence: junwang@hbut.edu.cn

Featured Application: The proposed algorithm for painting large objects based on a nine-axis UR5 robotic manipulator can be applicable in many automobile repair shops where paint jobs can be performed. With the help of a nine-axis UR5 robotic manipulator with the proposed algorithm, vehicles can be automatically painted with the least amount of human manual labor. Simultaneously, the quality and efficiency of the paint jobs can be drastically improved, since the UR5 robot maintains its consistency, accuracy, and proficiency while conducting paint jobs.

Abstract: An algorithm for automatically planning trajectories designed for painting large objects is proposed in this paper to eliminate the difficulty of painting large objects and ensure their surface quality. The algorithm was divided into three phases, comprising the target point acquisition phase, the trajectory planning phase, and the UR5 robot inverse solution acquisition phase. In the target point acquisition phase, the standard triangle language (STL) file, algorithm of principal component analyses (PCA), and k-dimensional tree (k-d tree) were employed to obtain the point cloud model of the car roof to be painted. Simultaneously, the point cloud data were compressed as per the requirements of the painting process. In the trajectory planning phase, combined with the maximum operating space of the UR5 robot, the painting trajectory of the target points was converted into multiple traveling salesman problem (TSP) models, and each TSP model was created with a genetic algorithm (GA). In the last phase, in conformity with the singularities of the UR5 robot's motion space, the painting trajectory was divided into a recommended area trajectory and a non-recommended area trajectory and created by the analytical method and sequential quadratic programming (SQP). Finally, the proposed algorithm for painting large objects was deployed in a simulation experiment. Simulation results showed that the accuracy of the algorithm could meet the requirements of painting technology, and it has promising engineering practicability.

Keywords: genetic algorithm; principal component analyses; standard triangle language; traveling salesman problem; trajectory planning



Citation: Wang, J.; Yang, M.; Liang, F.; Feng, K.; Zhang, K.; Wang, Q. An Algorithm for Painting Large Objects Based on a Nine-Axis UR5 Robotic Manipulator. *Appl. Sci.* **2022**, *12*, 7219. <https://doi.org/10.3390/app12147219>

Academic Editors: Pedro Neto, António Paulo Moreira and Félix Vilarinho

Received: 7 June 2022

Accepted: 14 July 2022

Published: 18 July 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

In recent years, UR5 robots have been widely popularized in industrial production fields such as painting, assembly, and micromanipulation [1]. In the above-mentioned fields, the painting process is the integral manufacturing procedure for automatically coating large objects, and it is one of the essential technologies for improving the surface quality of painted objects, which can then offer better performances under different working conditions [2–4]. Painting feasibility and trajectory planning are critical in the painting field.

To paint large objects, some engineers tried to mount UR5 robots on a mobile platform to enlarge the workspace of the UR5 robots [5,6], while other engineers tried to widen

the workspace by changing the structure of the UR5 robots [7]. However, the method of changing the structure of the UR5 robots entails enormous cost, and the UR5 robot with changed structure offers low adaptability. Thus, this paper proposes a UR5 robot which is mounted on an *zyz* three-axis motion platform to achieve the painting of large objects.

A well-designed trajectory can improve painting efficiency. Bureerat, S. et al. employed the MRPEIL-DE algorithm to optimize the trajectory of a six degree-of-freedom (DOF) UR5 robot [8]. Yin, S. et al. utilized mechanical learning methods to devise an energy-saving trajectory for industrial UR5 robots [9]. Serralheiro, W. et al. proposed employing a non-based trajectory planning method for time-energy optimization of a completely wheeled mobile UR5 robot [10]. Kazim, I.J. et al. compared improving the artificial potential field (APF) by the traditional particle swarm optimization (PSO) algorithm and the serendipity-based PSO (SBPSO) algorithm to control the path of a universal robot UR5 with collision avoidance [11]. Kazim, I.J. et al. utilized differential evolution (DE) optimization with the MATLAB toolbox, which has an applied robot operating system (ROS), to quantify the contour tracking performance of a collaborative universal manipulator robot (UR5) [12]. Al-Shanoon, A. et al. proposed a novel reliable framework for deep ConvNet combined with visual servoing using a single RGB camera [13]. Balanji, H.M. et al. proposed a novel calibration framework based on a single camera and computer vision techniques using ArUco markers [14]. Vivas, A. et al. designed the implementation of the control of a real UR5 robot from Matlab/Simulink using ROS [15]. Araki, R. et al. proposed a 6D pose estimation method for an object from a single RGB image for a robotic grasping task [16].

Nonetheless, most of the above-mentioned algorithms neither consider the singularities of the kinematics of the UR5 robot's joints nor meet the requirements of the painting process. Therefore, this paper proposes a painting algorithm which can generate a painting trajectory satisfying the painting of large objects. The painting algorithm employs the standard triangle language (STL) file, algorithm of principal component analyses (PCA), and *k*-dimensional tree (*k*-d tree) to create a digital model of the object to be painted. The digital model is converted into multiple traveling salesman problem (TSP) models, and each TSP model is created with a genetic algorithm (GA). The painting trajectory offers high precision and efficiency.

2. Construction of a Point Cloud Model of the Target Object

STL files are popular in computer graphics application systems, and their file formats are simple and widely harnessed in machine vision and 3D reconstruction [17,18].

2.1. STL File Parsing

A STL file consists of multiple triangles and can be divided into the American Standard Code for Information Interchange (ASCII) format and binary format as per the storage format. The STL file obtained in this paper was suitable for ASCII. The analysis of a STL file in the ASCII format is shown in Figure 1.

```

Solid<name>
Facet normal:  $n_i, n_j, n_k$ 
Outer loop
Vertex  $v_{1x}, v_{1y}, v_{3z}$ 
Vertex  $v_{1x}, v_{1y}, v_{3z}$ 
Vertex  $v_{1x}, v_{1y}, v_{3z}$ 
End loop
End facet
.....
End solid<name>

```

Figure 1. Parsing of a STL file in ASCII format.

In Figure 1, η_i , η_j , and η_k represent the x , y , and z components of the triangle patch normal vector, respectively; v_{ix} , v_{iy} , and v_{iz} represent the x , y , and z coordinates of the i -th triangular patch, respectively.

2.2. Filling Triangles

The STL file only contains the coordinates of the vertices of the triangles, and the coordinates of all points in the model cannot be obtained. To obtain the full 3-dimensional information of the model, the triangles must be filled. This paper utilized the depth-first search (DFS) to fill the triangles. The triangle is represented by S . The center of the triangle is represented by o . Lengths of the 3 sides of the triangle is represented by l_1 , l_2 , and l_3 . The triangle S is later separated into 3 smaller triangles S_1 , S_2 , and S_3 . The algorithm model and the main flowchart of the filling process are shown in Figures 2 and 3.

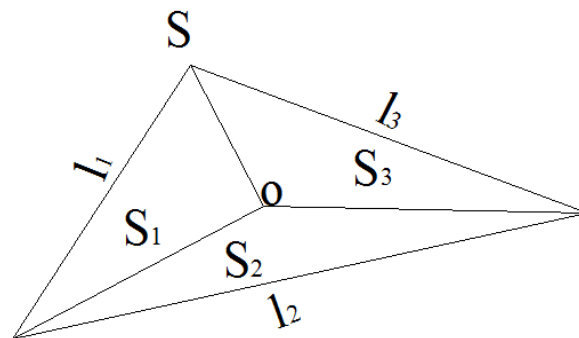


Figure 2. The model of the algorithm that fills triangles inside an STL file.

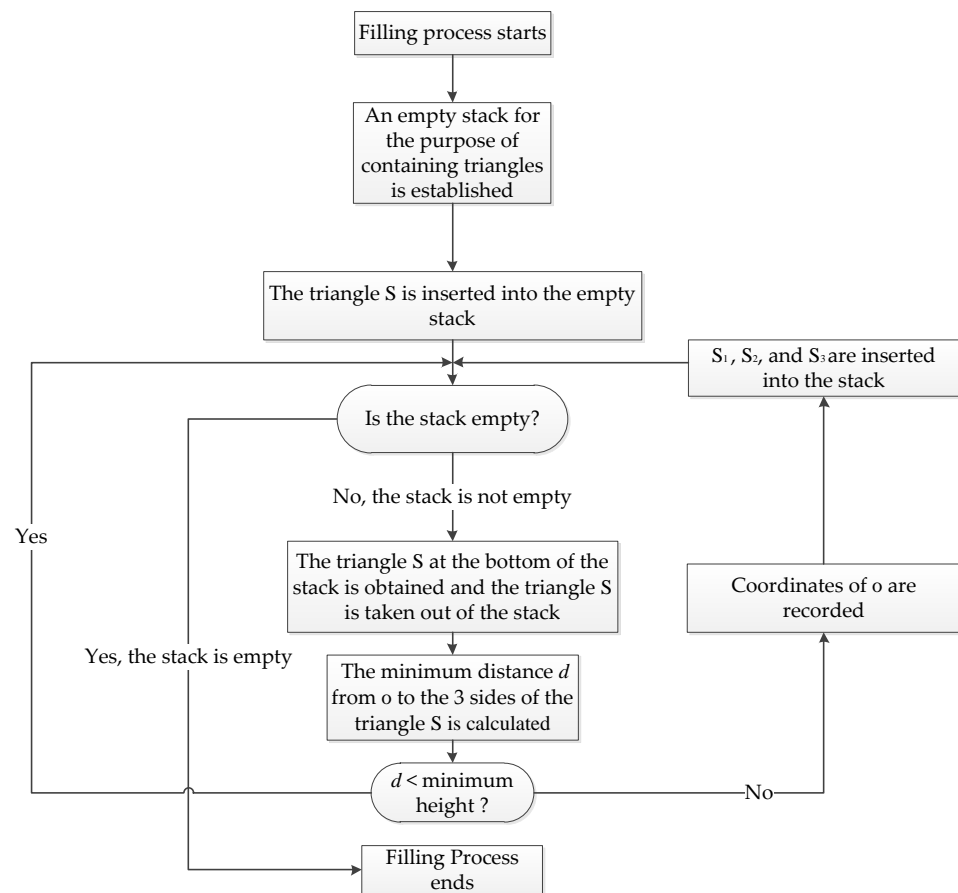


Figure 3. The filling process of the DFS algorithm.

The stopping condition of DFS recursion is that the minimum height corresponding to the three sides of the triangle is less than the specified minimum height. This requirement is shown in Equation (1), where the minimum triangle height h_{\min} is given.

$$S > \max(l_1, l_2, l_3) \cdot h_{\min} / 2 \quad (1)$$

The area of the triangle in Equation (1) is difficult to directly calculate. Therefore, Heron's formula, which is shown in Equation (2), is introduced, and the area is calculated from the lengths of the sides. The recursive stop condition is shown in Equation (3).

$$S = \frac{\sqrt{\varepsilon(\varepsilon - l_1)(\varepsilon - l_2)(\varepsilon - l_3)}}{\varepsilon = (l_1 + l_2 + l_3) / 2} \quad (2)$$

$$S \geq 2 \cdot \max(l_1, l_2, l_3) \cdot h_{\min} \quad (3)$$

The initial vertices are randomly selected as (10, 10, 0), (520, 100, 0), and (260, 410, 0), and h_{\min} equals 0.05. The filling results are shown in Figure 4a.

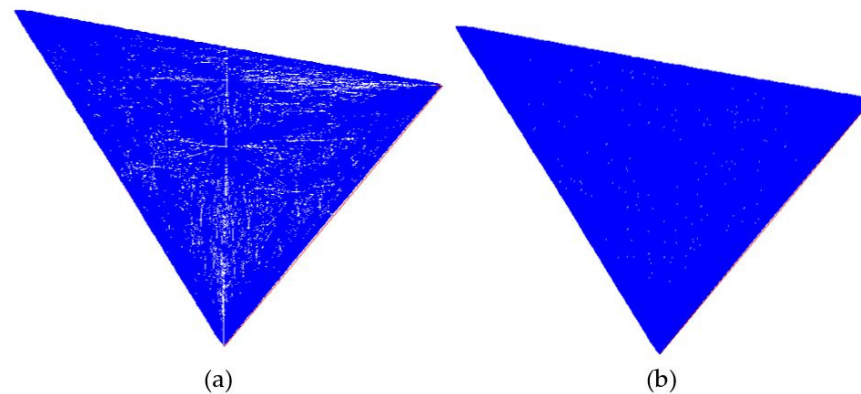


Figure 4. (a) The triangle filling results; (b) the improved triangle filling results.

However, there are a large number of unfilled lines in Figure 4a, and the filling effect is not satisfactory. After the filling process had been analyzed, it was found that the point filling method was harnessed in the filling, which could result in the points on the straight lines from the center of the triangle to the vertices of the triangle not being filled. If the number of filling points needs to be increased, the coordinates of the recorded point during the filling process can be changed into the coordinates of each point above the line connecting the recorded point and the vertex of the triangle with the same initial value condition and iteration termination condition. The final filling result is shown in Figure 4b. Compared to Figure 4a, Figure 4b is better filled, but the time taken to fill Figure 4b was much longer than to fill Figure 4a. If the accuracy is not high or the time is short, point filling is recommended. For high-precision conditions, linear filling is recommended.

2.3. Determining the Normal Vector of a Target Point on the Painting Trajectory

In order to ensure that the painting area is uniform during the painting process, it is necessary to ensure that the axis of the end of the spray gun coincides with the normal vector of the target point on the painting trajectory. Thus, the normal vectors of the target model must be obtained. As is shown in Figure 3, the target points on the painting trajectory can be divided into two types, which include the trajectory points obtained through the filling algorithm and the vertices of the original triangle.

The points obtained by filling are still coplanar with the original triangles. The normal vectors of the points obtained by filling can be determined with the normal vector of the plane. The normal vectors of the triangles can be directly extracted from the STL file. However, for the normal vectors of the vertices of the original triangle, there is a common vertex of multiple triangles. Ergo, the normal phase of the plane cannot be employed

instead of the normal phase of the points. In this paper, local plane fitting and principle component analysis (PCA) estimation methods were harnessed to create the normal vectors of the target points [19–21]. In order to acquire the target point field conveniently, the k-d tree algorithm was harnessed to store the coordinates of the target points [22]. The step-by-step algorithm is shown in Figures 5 and 6.

1. A local point that is less than r from the target point is taken.
2. A sphere is created by employing the local points with a target point as the origin of the 3 dimensional Cartesian System and the coordinates of a local point with the target point being the origin are x', y' and z' .
3. The tangent plane is fit with the local points. On the tangent plane: $\sqrt{n_x^2 + n_y^2 + n_z^2} = 1$.
4. The minimum sum of the squares of the distances between the normalized local points and the tangent plane is taken as the objective function.
5. Feature matrix S is constructed as per the objective function:

$$S = \begin{bmatrix} x_1' & x_1' & \dots & x_n' \\ y_1' & x_2' & \dots & x_n' \\ z_1' & x_3' & \dots & x_n' \end{bmatrix} \begin{bmatrix} x_1' & x_1' & \dots & x_n' \\ y_1' & x_2' & \dots & x_n' \\ z_1' & x_3' & \dots & x_n' \end{bmatrix}^T$$
6. The eigenvector corresponding to the minimum eigenvalue of the S matrix is taken as the normal vector of the points.

Figure 5. The PCA method to estimate the target point normal vector.

1. The variance of the target point set O in X, Y, Z dimensions is calculated.
2. The dimension k with the largest variance and the median p under the dimension are selected.
3. k, p are harnessed to divide O into O_1, O_2 .
4. If the number of element values in the set O_1, O_2 is greater than 1, the set O_1, O_2 is divided repeatedly.

Figure 6. The k-d tree stores cloud data.

In the actual processing procedure, the normal vector of the target model should take its external normal phase, and the external normal vector of the model can be determined by Equation (4), where \mathbf{p} is the center of all target points, and $\|x\|_2$ is the second norm of the vector x .

$$\mathbf{n} = \begin{cases} \mathbf{n} & \|(\mathbf{p} + \mathbf{n}) - \bar{\mathbf{p}}\|_2 \geq \|(\mathbf{p} - \mathbf{n}) - \bar{\mathbf{p}}\|_2 \\ -\mathbf{n} & \|(\mathbf{p} + \mathbf{n}) - \bar{\mathbf{p}}\|_2 < \|(\mathbf{p} - \mathbf{n}) - \bar{\mathbf{p}}\|_2 \end{cases} \quad (4)$$

2.4. Determining the Pose of a Target Point on the Trajectory

In the actual painting process, the pose matrix is often deployed to describe the rotation of the target point in space. During the inverse kinematics calculation of the UR5 robot, the pose of the target point needs to be described by the matrix, and Rodrigues' rotation formula can determine the pose of the target point by the axis direction of the target point.

The diagram of Rodrigues' rotation formula is shown in Figure 7, where $X_1, Y_1,$ and Z_1 represent the three axes, with O being the origin of the 3-dimensional Cartesian system after rotating.

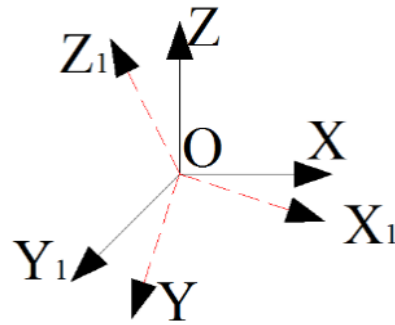


Figure 7. The schematic diagram of Rodrigues' rotation formula.

The rotation matrix R is shown in Equation (5),

$$E \cos \theta + (1 - \cos \theta)r^T \times r + \sin \theta \times \begin{bmatrix} 0 & -r_z & r_y \\ r_z & 0 & -r_x \\ -r_y & r_x & 0 \end{bmatrix} \tag{5}$$

where E represents the third-order identity matrix, and θ represents the angle created by vectors \mathbf{n} and \mathbf{n}_0 . In Equation (5), \mathbf{r} is the vector product of vectors \mathbf{n} and \mathbf{n}_0 ; $\mathbf{n}_0 = [0 \ 0 \ 1]^T$, and \mathbf{n} is the normal vector of the point.

2.5. Compression of the Data of Target Points

The design of the painting process parameters ensures that the end of the spray gun will produce a circle with a radius of r at a specified distance. During the painting process, it is necessary to ensure that the trajectory coverage during painting is 0.4. The painting model is shown in Figure 8, and the coverage of Figure 8, denoted by p , is shown in Equation (6).

$$p = \frac{4r^2 \cdot \arccos(d/2r) - d\sqrt{r^2 - d^2/4}}{2\pi r^2} \tag{6}$$

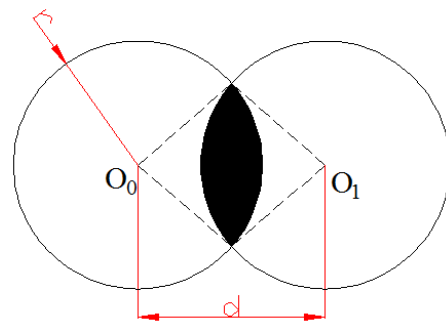


Figure 8. The painting model.

When the painting trajectory is executed, not all target points are to be painted. Only the path of key points should be painted. The distance between key points should meet the requirements of painting uniformity. In Figure 8 and Equation (6) d is employed to reduce the time and space complexity of the trajectory planning. The compression formula for the data points is shown in Equation (7).

$$\begin{aligned} k &= \text{floor}(d/(\sqrt{3})) \\ P_x &= \text{round}(P_x/k) \cdot k \\ P_y &= \text{round}(P_y/k) \cdot k \\ p_z &= \text{round}(p_z/k) \cdot k \end{aligned} \tag{7}$$

where P_x , P_y , and P_z represent the x - y - z coordinates of the target points in the point cloud, $floor$ represents the floor function, which takes as input a real number q , and gives out as output the greatest integer less than or equal to q , and $round$ represents the round function, which rounds off a numeric value to its nearest integer.

3. Nine-Axis UR5 Robot Forward Kinematics Model

The space that the UR5 robot covers is limited, and large-size objects may not be painted at once. When an area that needs to be painted is outside the working space of the UR5 robot, the D-H parameters of the UR5 robot must be changed, or auxiliary equipment must be added to help with the painting. The former is costly. Changing joints is not conducive to the popularization of UR5 robots. The proposed 6-DOF UR5 robot is installed on a 3-axis motion platform that moves horizontally, laterally, and vertically to assist painting. The 3D drawing of the UR5 robot is shown in Figure 9. The 9-axis UR5 robotic manipulator includes the 3-axis motion platform that has 3 prismatic pairs and the UR5 robot that has 6 revolute pairs. The 6-DOF UR5 robot selected in this paper is a UR5 robot, which is shown in Figure 10.

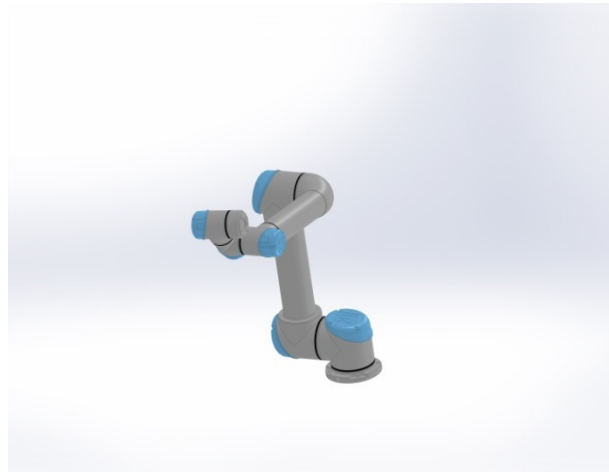


Figure 9. 3D drawings of the UR5 robot.

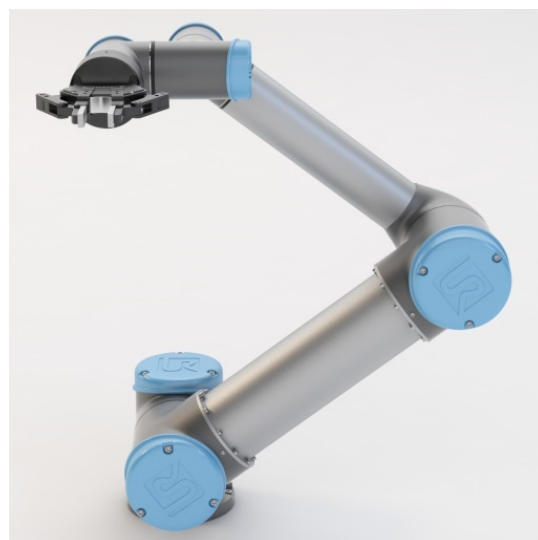


Figure 10. The UR5 robot.

The pose of the actual UR5 robot end \mathbf{T} is shown in Equation (8).

$$\mathbf{T} = \mathbf{Trans}(t_x, t_y, t_z)\mathbf{Rot}(x, \pi)\mathbf{T}_{\text{robot}}\mathbf{T}_{\text{tool}}\mathbf{T}_{\text{dis}}$$

$$\mathbf{Trans}(t_x, t_y, t_z) = \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{8}$$

$$\mathbf{Rot}(x, \pi) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos \pi & \sin \pi & 0 \\ 0 & -\sin \pi & \cos \pi & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

in which t_x , t_y , and t_z represent the movement of the motion platform along the x , y , and z axes, respectively, and \mathbf{Trans} represents the movement operator. \mathbf{Rot} represents the rotation matrix of the x -axis. \mathbf{T}_{tool} , which is shown in Figure 11, represents the pose of the end-effector relative to its installation center. \mathbf{T}_{dis} , which is also shown in Figure 11, represents the end fixture of the UR5 robot's position. \mathbf{T}_{tool} is determined by the structure of the end-effector, and \mathbf{T}_{dis} is determined by the parameters of the end fixture of the UR5 robot's position. The UR5 robot system has 9 axes which are described in Figure 11. Axes 1 to 6 are the 6 joints of the UR5 robot. Axes 7 to 9 are the three-dimensional Cartesian coordinate system of the motion platform.

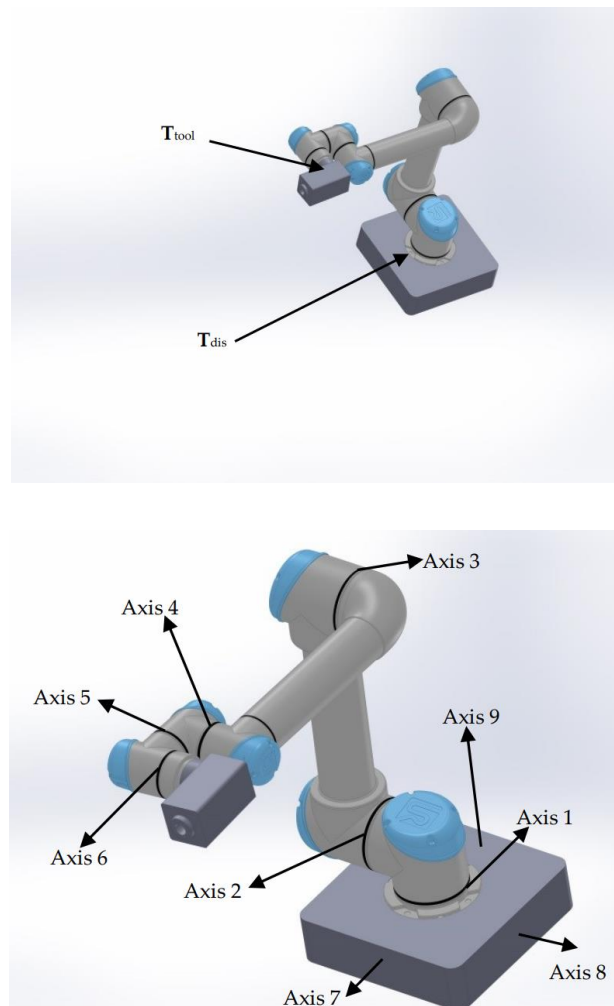


Figure 11. \mathbf{T}_{tool} , \mathbf{T}_{dis} , and the 9 axes of the UR5 robot system.

Figure 12a is the structural diagram of the 6-DOF UR5 robot composed of 6 revolute pairs, represented by 1, 2, 3, 4, 5, and 6 and 0 represents the end fixture of the UR5 robot. In order to analyze the pose change of the UR5 robot’s end fixture coordinates relative to the end-effector, the D-H model is harnessed to establish the kinematics forward solution model of the UR5 robot.

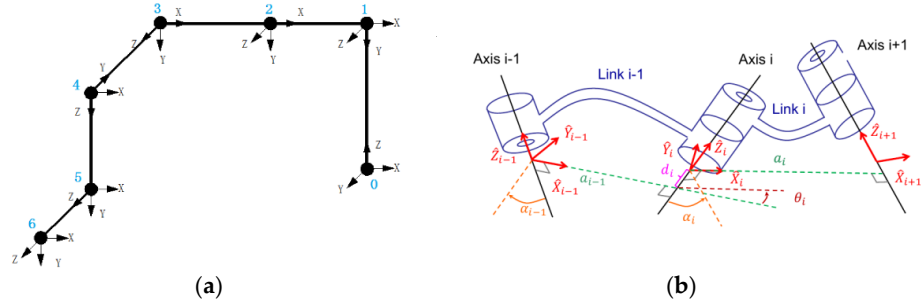


Figure 12. (a) The schematic structure of the UR5 robot (b) detailed explanation of θ_i , d_i , a_i , and α_i .

The UR5 robot’s pose change matrix in the D-H model is shown in Equation (9).

$${}^i{}_{i-1}\mathbf{T} = \mathbf{Rot}(z, \theta_i)\mathbf{Trans}(0, 0, d_i)\mathbf{Trans}(a_i, 0, 0)\mathbf{Rot}(x, \alpha_i)$$

$$\mathbf{Rot}(z, \theta_i) = \begin{bmatrix} \cos \theta_i & \sin \theta_i & 0 & 0 \\ -\sin \theta_i & \cos \theta_i & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{9}$$

In Equation (9), θ_i is the angle for which x_{i-1} spins around z_i to become x_i ; d_i is the distance between x_{i-1} and x_i along z_i ; a_i is the distance between z_i and z_{i+1} along x_i , and α_i is the angle for which z_i spins around x_i to become z_{i+1} . $\mathbf{Rot}(z, \theta_i)$ represents the rotation matrix of the z -axis; θ_i , d_i , α_i , and a_i are shown in Figure 12b. ${}^i{}_{i-1}\mathbf{T}$ is shown in Equation (10), where s represents the sine function and c represents the cosine function.

$${}^i{}_{i-1}\mathbf{T} = \begin{bmatrix} c\theta_i & -s\theta_i c\alpha_i & s\theta_i s\alpha & a_i c\theta_i \\ s\theta_i & c\theta_i c\alpha_i & -c\theta_i s\alpha & a_i s\theta_i \\ 0 & s\alpha_i & c\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{10}$$

$\mathbf{T}_{\text{robot}}$, which represents the pose of the UR5 robot, is obtained via Equation (11). The UR5 robot’s pose should satisfy Equation (12), and the D-H parameters of the UR5 robot are shown in Table 1.

$$\mathbf{T}_{\text{robot}} = \prod_{i=1}^6 {}^i{}_{i-1}\mathbf{T} \tag{11}$$

$$\mathbf{Trans}(x, y, z)^{-1} \cdot \mathbf{Rot}(x, \pi)^{-1} \cdot \mathbf{T} \cdot \mathbf{T}_{\text{tool}}^{-1} = \prod_{i=1}^6 {}^i{}_{i-1}\mathbf{T} \tag{12}$$

Table 1. The D-H parameters of the UR5 robot.

i	θ_i	d_i/mm	α_i	α_i/mm
1	-2π to 2π	89.2	$\pi/2$	0
2	-2π to 2π	0	0	425
3	-2π to 2π	0	0	392.3
4	-2π to 2π	109.2	$\pi/2$	0
5	-2π to 2π	94.7	$-\pi/2$	0
6	-2π to 2π	82.3	0	0

4. The Inverse Solution Model of Kinematics of the 9-Axis UR5 Robot

Nevertheless, since the system that controls the UR5 robot and the system that controls the motion platform are separated, communication between the systems may cause problems, such as delays and errors. In order to improve painting accuracy, the motion platform cannot be moved frequently. During the painting process, each time the motion platform is moved, the UR5 robot paints an area. After the painting is completed, the motion platform is moved again. The recommended operating space of the UR5 robot is shown in Figure 13.

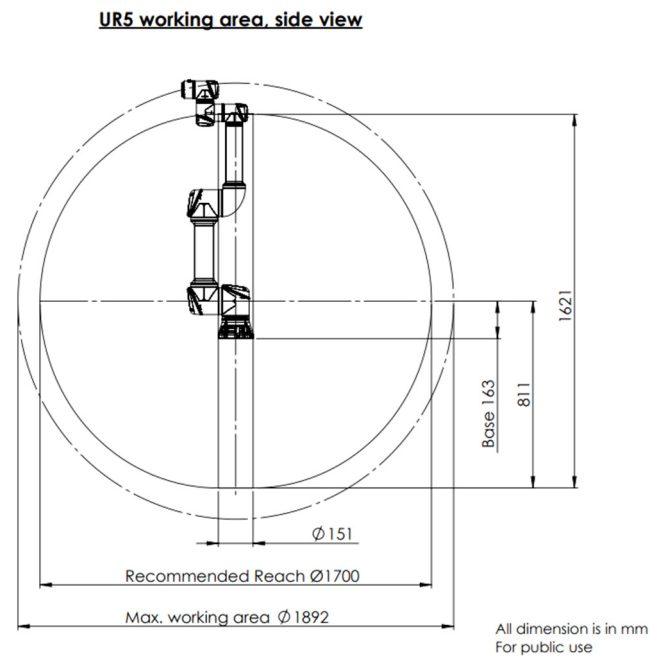


Figure 13. The working area of the UR5 robot.

The recommended working area in Figure 13 represents the feasible area recommended by the UR5 robot. The area between the maximum working area, which is represented by Max. working area in Figure 13, and the recommended feasible area represents the non-recommended feasible area that the UR5 robot may not reach. The non-recommended area represents the singularity of the UR5 robot. The non-recommended area takes the largest cylinder contained in the green area in Figure 13 as the largest area for each painting procedure.

However, the recommended area in Figure 13 is not a complete ball, and the cylindrical area with a diameter of 151 mm is not recommended. If this area is removed, the volume of the largest cylinder obtained is 0.25 times that without removal. Therefore, the platform will be moved slightly. In order to move the platform as little as possible, the green area is assumed to be a complete ball when the maximum cylinder is calculated.

In order to prevent the UR5 robot from colliding with the platform, it is necessary to keep the end of the UR5 robot below its installation center during the movement of the UR5 robot. The area under the side wall of the machine is taken as the effective area. The volume of the cylinder included in Figure 13 is shown in Equation (13).

$$V = \pi(r\cos\theta)^2 \cdot (163 - r\sin\theta) \tag{13}$$

where r , θ , and V represent the maximal length of the UR5 robot, the radian of the intersection of the cylinder, and the maximum machining space of the UR5 robot, respectively. When θ equals -0.54 rad, the volume of the cylinder is the largest. At this time, the cylinder has a radius of 727 mm and a height of 602 mm. Due to the existence of the non-recommended area, the UR5 robot inverse kinematics model is divided into a recommended area inverse kinematics model and a non-recommended area inverse kinematics model.

4.1. Inverse Solution Model for Recommended Regions

The three joint axes of the UR5 robot, joint axes 2, 3 and 4, which are shown in Figure 14, are parallel, and their spatial structure satisfies the Pieper criterion. The closed-form solution method can be harnessed to solve the angular sequence of the UR5 robot joint under the target pose constraints.

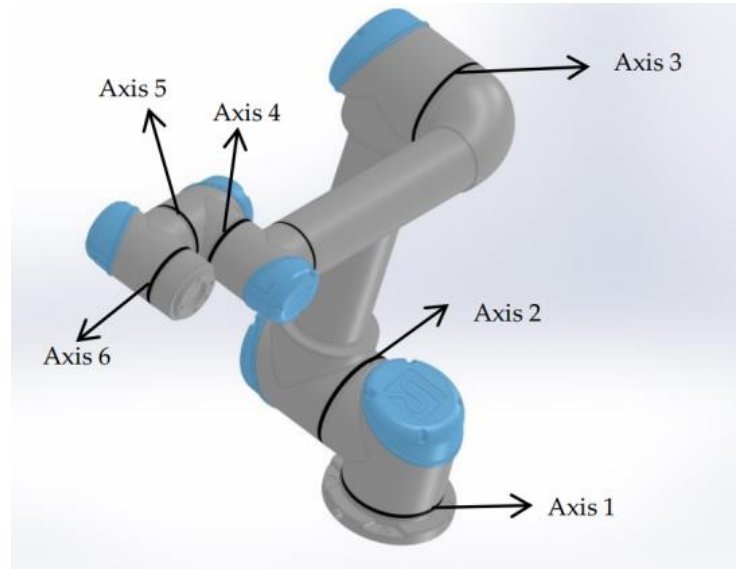


Figure 14. The axes of the UR5 robot.

Equation (14) determines the pose of the end-effector of the UR5 robot.

$$\prod_{i=1}^6 {}^{i-1}\mathbf{T} = (\mathbf{Trans}(t_x, t_y, t_z)\mathbf{Rot}(x, \pi))^{-1} \cdot \mathbf{T} \cdot (\mathbf{T}_{tool} \cdot \mathbf{T}_{dis})^{-1} = \begin{bmatrix} n_x & o_x & a_x & x' \\ n_y & o_y & a_y & y' \\ n_z & o_z & a_z & z' \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (14)$$

The main idea of the closed-form solution is creating the constraint equation established by matrix changes. The constraint matrix of the UR5 robot, which is shown in Equation (15), is established in accordance with Equation (11). The third row of the left and right sides of Equation (15) is expanded to establish Equation (16). Equation (16) can be regarded as the equation set of θ_1 , θ_5 , and θ_6 .

The procedure for finding the values of θ_2 , θ_3 , and θ_4 is similar.

$$\mathbf{L} = ({}^0\mathbf{T}^{-1})\mathbf{T}_{robot} = \prod_{i=2}^6 {}^{i-1}\mathbf{T} = \mathbf{R} \quad (15)$$

$$\begin{bmatrix} c\theta_6s\theta_5 & & & \\ s\theta_6s\theta_5 & & & \\ c\theta_5 & & & \\ d_2 + d_3 + d_4 + d_6c\theta_5 + a_5s\theta_5 + a_6c\theta_6s\theta_5 & & & \end{bmatrix}^T = \begin{bmatrix} n_y c\theta_1 + n_x s\theta_1 \\ o_y c\theta_1 + o_x s\theta_1 \\ a_y c\theta_1 + a_x s\theta_1 \\ y c\theta_1 + x s\theta_1 \end{bmatrix}^T \quad (16)$$

By employing Equations (15) and (17), the result can be obtained in Equation (18),

$$\mathbf{L} = ({}^0\mathbf{T}^{-1}){}^0\mathbf{T}({}^4\mathbf{T}^{-1})({}^5\mathbf{T}^{-1}) = \prod_{i=2}^4 {}^{i-1}\mathbf{T} = \mathbf{R} \quad (17)$$

$$\mathbf{R} = \begin{bmatrix} c\theta_{234} & 0 & -s\theta_{234} & a_3 \cdot c\theta_{23} + a_3 \cdot c\theta_2 + a_4 \cdot c\theta_{234} \\ -s\theta_{234} & 0 & c\theta_{234} & -a_3 \cdot s\theta_{23} - a_3 \cdot c\theta_2 - a_4 \cdot c\theta_{234} \\ 0 & -1 & 0 & d_2 + d_3 + d_4 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{18}$$

where $\theta_{i_1 i_2 \dots i_n}$ is defined in Equation (19).

$$\theta_{i_1 i_2 \dots i_n} = \sum_{j=1}^n \theta_{ij} \tag{19}$$

In Equation (18), θ_{234} , θ_{23} , and θ_2 can be acquired by the fourth column of \mathbf{R} in Equation (17); θ_2 , θ_3 , and θ_4 can be obtained afterwards. In Equations (16) and (18), s represents the sine function and c represents the cosine function. Nonetheless, the solution involves a large number of inverse trigonometric functions, and the range of the angles within which the UR5 robot’s joints move is -2π to 2π , which results in multiple solutions. Therefore, the solution with the smallest Euler distance from the initial joint angle sequence is selected.

4.2. The Inverse Solution Model for Non-Recommended Regions

In a non-recommended area, the UR5 robot may not be able to achieve certain poses. It is necessary to move the motion platform to match the position of the UR5 robot. Then, the inverse solution model of the manipulator can be changed to obtain the minimum joint rotation radian and the number of movements of the manipulator’s motion platform under posture and pose constraints. This problem is a nonlinear optimization problem with constraints. Sequential quadratic programming (SQP) is an iterative method for constrained nonlinear optimization. When the input value is close to the real solution, the algorithm has a second-order convergence speed and can quickly solve the target solution [23,24].

4.2.1. The Objective Function and Constraints

In the process of solving the inverse kinematics, in order to avoid singularities, the platform is allowed to move slightly. The number of movements of the platform and the total arc of the UR5 robot joint give the objective function, which is shown in Equation (20).

$$f = \sqrt{\sum_{i=1}^{i=6} (\theta_i^r - \theta_i)^2 + K_d \left((x^r - t_x)^2 + (y^r - t_y)^2 + (z^r - t_z)^2 \right)} \tag{20}$$

where θ_i^r and θ_i are the actual joint curvature and the starting point arc of the UR5 robot. The target point K_d represents the number of movements of the platform; x^r , y^r , and z^r represent the actual moving distance.

The error es is allowed in the actual working situation. The constraint condition should be that the Euler distance between \mathbf{T} and the target pose \mathbf{T}' is less than es , which is shown in Equation (21).

$$\sqrt{\sum_{j=1}^{j=3} \left({}^0\mathbf{T}_{j4} - {}^0\mathbf{T}'_{j4} \right)^2} - es \leq 0 \tag{21}$$

During the painting process, an error es' between the end axis of the spray gun and the target point axis is allowed. The pose constraints are shown in Equation (22).

$$\left| \arccos \left(\frac{\mathbf{n} \cdot \mathbf{n}_0}{|\mathbf{n}| \cdot |\mathbf{n}_0|} \right) \right| < es' \tag{22}$$

where \mathbf{n} is the normal direction of \mathbf{T} to the z axis, and \mathbf{n}_0 is the normal direction of \mathbf{T}' to the z axis.

4.2.2. The Solution Process of the SQP Algorithm

The SQP algorithm decomposes the problem into the quadratic programming (QP) sub-problem, obtaining the descending direction d by solving the current angle sequence θ_k in the QP problem and updating θ_{k+1} via d until the Karush–Kuhn–Tucker (KKT) condition is satisfied or the maximum number of iterations is reached. The specific solving process of the algorithm is described below.

(1) The Lagrange function $L(\theta_k, \lambda)$ of the current angular sequence θ_k is constructed, and the expression of $L(\theta_k, \lambda)$ is shown in Equation (23).

$$L(\theta_k, \lambda) = f(\theta_k) + \lambda g(\theta_k) \quad (23)$$

where λ is the Lagrangian multiplier, and $\lambda \geq 0$.

(2) The descending direction d of the current angular sequence θ_k is obtained. Equation (23) is converted into the solution to d , which is shown in Equation (24).

$$\begin{aligned} \min_d \quad & \nabla f(\theta_k)^T d + \frac{d^T H_k d}{2} \\ \text{st} : \quad & g(\theta_k) + \nabla g(\theta_k)^T d \leq 0 \end{aligned} \quad (24)$$

In Equation (24), H_k represents the Hessian matrix of the Lagrangian function $L(\theta_k, \lambda)$.

(3) The angle sequence in which θ_{k+1} equals θ_k plus d is updated, and whether the result of the solution satisfies the KKT condition or reaches the maximum number of iterations is determined. If the KKT condition is satisfied, the iteration is stopped. Step 2 is repeated if the KKT condition is not satisfied. The KKT condition is shown in Equation (25) [25].

$$\begin{aligned} \nabla f(\theta_k) + \lambda g(\theta_k) &= 0 \\ \lambda &\geq 0 \\ g(\theta_k) &= 0 \\ \lambda g(\theta_k) &= 0 \end{aligned} \quad (25)$$

In the KKT condition, the third condition means that the result of the solution satisfies the constraint. If $g(\theta_k)$ equals 0, the KKT condition becomes $\nabla f(\theta_k)$, which equals 0. If $g(\theta_k)$ is smaller than 0 and λ equals 0, the KKT condition also becomes $\nabla f(\theta_k)$, which equals 0.

5. The Painting Trajectory

During the painting process, not only should the trajectory of the motion platform be planned, but the trajectory of the UR5 robot's movement should also be planned. During the painting process, the posture of the UR5 robot end is shown in Equation (26).

$$\mathbf{Trans}(x, y, z) \cdot \mathbf{Rot}(x, p) \cdot \mathbf{T}_{robot} = \mathbf{T} \cdot \mathbf{T}_{dis}^{-1} \cdot \mathbf{T}_{tool}^{-1} \quad (26)$$

Since $\mathbf{Rot}(x, \pi)$ only affects the directions of the x -axis and z -axis of the 3-dimensional Cartesian system of the UR5 robot system, it does not change the range of the UR5 robot's operating space. Equation (26) can be transformed into Equation (27).

$$\mathbf{Trans}(x, y, z) \cdot \mathbf{T}_{robot}' = \mathbf{T} \cdot \mathbf{T}_{dis}^{-1} \cdot \mathbf{T}_{tool}^{-1} \quad (27)$$

where $\mathbf{T}'_{UR5\ robot}$ equals $\mathbf{Rot}(x, \pi) \cdot \mathbf{T}_{UR5\ robot}$.

5.1. The Trajectory of the Motion Platform

When the size of the object to be painted exceeds the maximum painting area of the UR5 robot, the auxiliary movement of the motion platform is required to complete the painting. During the painting process, the motion platform is moved to bring the UR5 robot close to the target area, and the UR5 robot paints the area. After painting, the motion platform is moved again, and the UR5 robot paints another area. Then, the trajectory

planning of the motion platform is optimized by taking the least number of movements of the motion platform as the objective function.

Because the UR5 robot’s permitted painting space is a cylinder and the height of the painted object is generally less than the maximum height of the painting space, the model can be projected onto the xy plane. The model can be transformed to cover all target points with the fewest circles.

5.1.1. The Minimum Envelope Rectangle of the Target Points

Since the number of target points is large, in order to simplify the calculation, a rectangle enveloping the target points—instead of a set of target points—is harnessed. The solution model, which is shown in Figure 15, is created by employing the minimum rectangle method of the main direction target points. The dotted section in the figure represents the target points; the blue line represents the main direction of the target points, and the red rectangle represents the minimum envelope rectangle of the target points.

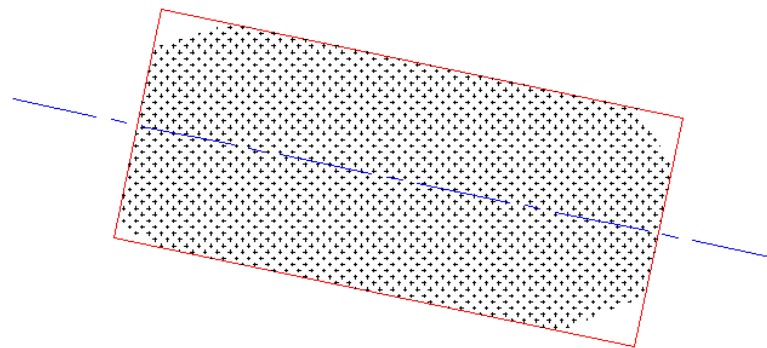


Figure 15. The minimum envelope rectangle of the target points.

The method for obtaining the principal direction of the plane is shown in Equation (28).

$$\theta = \frac{\text{atan2}(2M_{11}, M_{20} - M_{02})}{2} \tag{28}$$

in which θ is the minimum angle between the major axis direction and the direction of the positive side of the x -axis, and M_{pq} is the $p + q$ order center moment of the projection surface. M_{pq} is shown in Equation (29).

$$M_{pq} = \sum_x \sum_y (x - \bar{x})^p (y - \bar{y})^q \tag{29}$$

where \bar{x} and \bar{y} represent the center coordinates of the original target point.

5.1.2. The Minimum Number of Movements of the Motion Platform

When a circle fills a rectangle, the effective filling area of each circle is generally rectangular. Then, the model can be simplified again by filling the target rectangle with the smallest number of rectangles inside the circle. The filling model is generally shown in Figure 16. The smaller rectangle in the figure represents the target rectangle that needs to be filled. The larger rectangle represents the largest filled area, and the circular areas outside the largest rectangle represent the areas not recommended for use by the UR5 robot. Each circle represents the maximum processing range of the system each time. P_x and P_y represent the vertical distance and horizontal distance between the target rectangle and the largest filled area; l and h represent the length and width of the effective rectangle inside each circle.

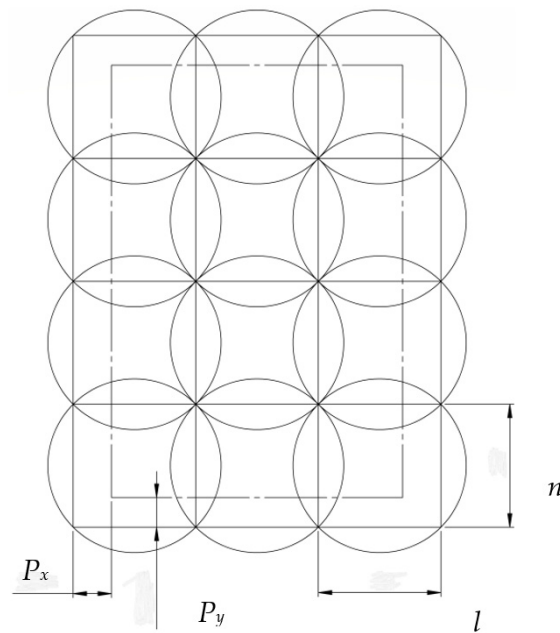


Figure 16. The schematic diagram of the filling model.

The size of the rectangle inside the circle indicates the effective area size of the circle. The length and width of the maximum inscribed rectangle in the circle are both $\sqrt{2}r$. The length and width of the inscribed rectangle of the circle are generally close to $\sqrt{2}r$ when the target rectangle is selected. The constraints of the filled model are shown in Equation (30).

$$\begin{aligned}
 n &= \left[l_{\max} / \sqrt{2}r \right] + 1 \\
 \sum_{i=1}^n l_n &\geq l_{\max} \\
 \sqrt{4r^2 - l_i^2} &\geq h_{\max}
 \end{aligned}
 \tag{30}$$

where $[l_{\max} / \sqrt{2}r]$ represents the largest integer that does not exceed $\frac{l_{\max}}{\sqrt{2}r}$.

Since the number of circles in Figure 16 is an integer, there are many kinds of P_x , P_y , h_1 , and h_2 satisfying Equation (29). However, the UR5 robot’s non-recommended area may cause the movement of the motion platform. The minimum number of target points in the non-recommended area during processing is the optimization goal, and the optimal P_x , P_y , h_1 , and h_2 are obtained by employing the exhaustive method.

5.1.3. The Movement Sequence of the Motion Platform

Reasonably planning the movement sequence of the motion platform can reduce the movement of the motion platform. During processing, the initial point of the motion platform is at zero. The movement sequence planning model of the motion platform can be transformed into a TSP model starting from zero and returning to zero after accessing all target circle centers. Because the number of points is low, the problem lies in creating a small TSP model. The problem can be directly solved by employing the example method. Since each axis of the motion platform is controlled by a separate motor, the Halton distance between two points is taken as the weight when creating the TSP model.

5.2. The Kinematic Trajectory of the UR5 Robot

According to Figure 8, some painting areas overlap, and, therefore, the greedy principle is harnessed to include as many target points as possible for each time of painting. The painting process requires uniform motion during the painting process, and, therefore, the shorter the total painting path is and the shorter the painting time is, the higher the painting efficiency is.

When each area is painted, the painted model can be transformed into a TSP model that finds the shortest path which can access all of the target points [26]. When different areas are painted, in order to prevent the UR5 robot from colliding with the processed objects, the UR5 robot’s joints must be moved after returning to the origin of the 3-dimensional Cartesian system. Then, the UR5 robot kinematics trajectory planning model can be transformed into the solution of multiple TSP models.

However, due to the large number of points to be painted, this model is a large TSP model. Genetic algorithms (GA) are harnessed to create each TSP model [27,28].

5.2.1. The TSP Model Based on the Genetic Algorithm

A genetic algorithm-based computational model that simulates the evolutionary process of natural selection and genetic mechanisms of Darwin’s theory of evolution is proposed. The genetic algorithm does not need continuous function limitation and has a better global optimal solution. The core method of GA is to evaluate the fitness of all individuals in each generation of the population and select some individuals for genetic selection, crossover, and mutation to form the next generation population. The main solution steps of the genetic algorithm are shown in Figure 17.

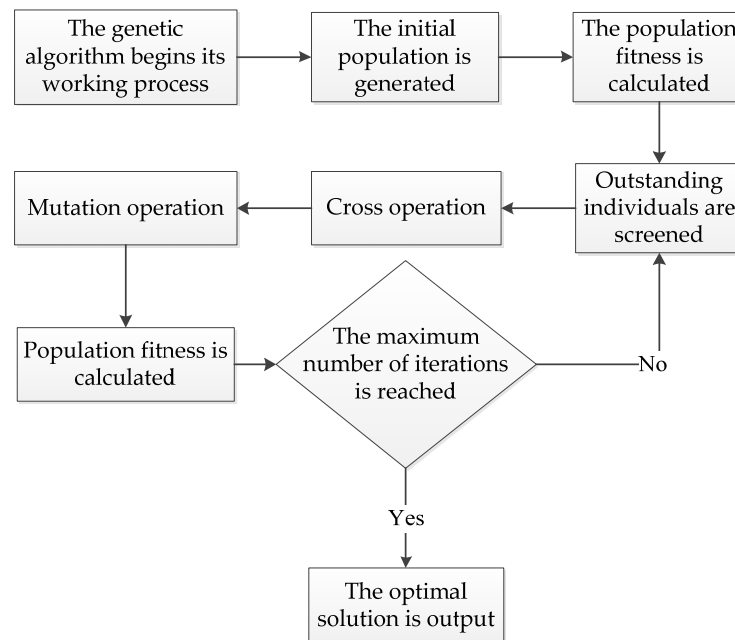


Figure 17. The solution steps of the genetic algorithm.

5.2.2. Operations Related to the Genetic Algorithm

(1) Coding method: all target points are encoded into one chromosome $G_1G_2 \dots G_wG_k$, and G_i in the chromosome represents the i -th number of the target point. In this genetic algorithm, each individual has one and only one chromosome.

(2) Weight: the Euler distance between two points is taken as the weight between the two points.

(3) Chromosome distance and individual fitness: the fitness of the individual f equals $N / \sum_{i=2}^n d_{i(i+1)}$, where N represents the gain coefficient. The value of N only affects the value of fitness and does not affect the final TSP solution result.

(4) Selection operation: the fitness of the population is calculated; the best individual is screened, and the coding method of the individual is retained. For the remaining individuals, the individual distribution function is calculated in accordance with their individual fitness. The maximum and minimum normalization algorithms are employed

to normalize the distribution function to 0-1. The selection operation process is shown in Figure 18.

- 1: An empty population vector of n is created.
- 2: The population records the best optimal individual
- 3: A random number of 0-1 is generated.
- 4: The maximum number of points is found which is less than or equal to the random number in line with the distribution function
- 5: The population records the points' numbers
- 6: If the size of the population is less than n , steps 3, 4, and 5 are repeated.
- 7: The population is output.

Figure 18. The selection operation process.

(5) Crossover operation: in order to ensure the positive optimization of the genetic algorithm, crossover operations are performed only on non-optimal individuals in the population. The crossover operator uses the Order Crossover operator. The crossover algorithm is shown in Figure 19, where P_c represents the crossover probability.

- 1: Two individual parent 1 and parent 2 are selected in the population in order. Their individual chromosomes are $G_1G_2...G_n$ and $G_nG_{n-1}...G_1$.
- 2: A random P number within 1 is generated.
- 3: If $P > P_c$, parent 1 and parent 2 are output and the cross operation of the individual is finished.
- 4: Cross positions *start* and *end* are randomly generated.
- 5: Individual chromosomal gene fragments are generated:
offspring 1: ... $G_{start}...G_{end}$... and offspring 2: ... $G_{end}...G_{start}$
...
- 6: The unknown gene fragments in offspring1 are filled in conformity with the genetic order of the parent 2 and the unknown gene fragments in offspring 2 are filled as per the genetic order of the parent1. The genes in the offspring are not duplicated during the filling process. For example, if $n = 5$, $start = 2$ and $end = 3$, offspring 1's chromosomal is $G_5G_2G_3G_4G_1$ and offspring 2's chromosomal is $G_1G_4G_3G_2G_5$.
- 7: Offspring 1 and offspring 2 are output.

Figure 19. The crossover algorithm.

(6) Mutation operation: in order to prevent the GA from falling into a local optimal solution, an adaptive mutation rate is harnessed. The equation for calculating the mutation probability is shown in Equation (31),

$$P_m = \begin{cases} \frac{K_1(f_{\max} - f)}{f_{\max} - f_{avg}}, & f \geq f_{avg} \\ k_2, & f < f_{avg} \end{cases} \quad (31)$$

where f_{\max} represents the maximum fitness of the group; f is the fitness of the individuals to be crossed. K_1 and K_2 are the adaptive parameters. The mutation operation is shown in Figure 20.

- 1: One individual in the population in order is selected. The individual's chromosome is $G_1G_2\dots G_n$
- 2: A random P number within 1 is generated.
- 3: If $P > P_c$, this individual is output and the cross operation of the individual is finished.
- 4: Cross positions *start* and *end* are randomly generated.
- 5: The genes are reversed from start to end in an individual and the chromosome of this individual will become $G_1\dots G_{start-1}G_{end} \dots G_{start}G_{end+1}\dots G_n$
- 6: This individual is output.

Figure 20. The mutation operation.

6. Simulation Experiments

The spray gun of model WA-101 was adopted as the painting equipment, and the offset distance of the spray gun from the UR5 robot end axis was $T_{tool} = \mathbf{Trans}(5.5, 94.5, 165.5)$.

The painting process requirements are described in this paragraph. The distance between the fixture and the target point was T_{dis} , which equals $\mathbf{Trans}(0, 0, 113)$, and the end of the spray gun generated a circle with a radius of 50 mm at this distance. The maximum position error of the target point was 5 mm, allowing a 5° error between the gun's end axis and the target's normal vector.

6.1. The Pose Acquisition of the Target Objects

The analysis of the STL file of a car is shown in Figure 21, and the point cloud model of the car is shown in Figure 22. The shapes and sizes of Figures 21 and 22 are the same as the actual three-dimensional model. The target extraction algorithm proposed in this paper has promising practicality.

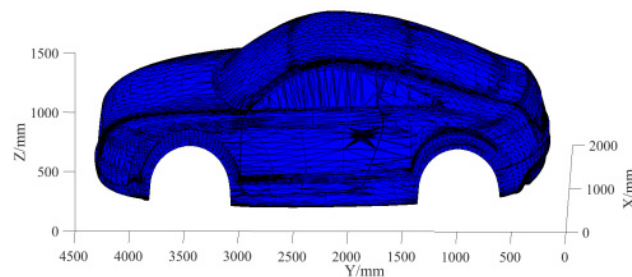


Figure 21. The STL analysis model of a car.

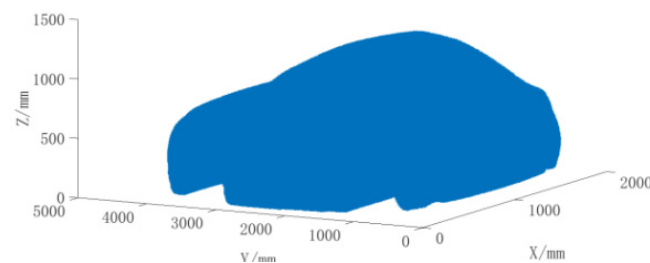


Figure 22. The point cloud model of the car.

The part of the car roof where z coordinates are larger than 1000 mm was taken for the target points for painting. The normal vector at the end of the target point was created by employing the PCA algorithm, as is shown in Figure 23 (because the performance of the computer graphics card was not satisfactory, and only the normal phase of some points is shown in the figure). As can be seen from Figure 23, the normal vectors of the target points conform to the shape of the roof, and the algorithm can better establish the normal vectors

of the target points. After the variable r in Equation (6) had been given a value of 50, the model of the compression points of the car roof, which is shown in Figure 24, was achieved as per Equation (7). After Figures 23 and 24 are compared, it is clear that the size and shape of the target points remain unchanged after compression.

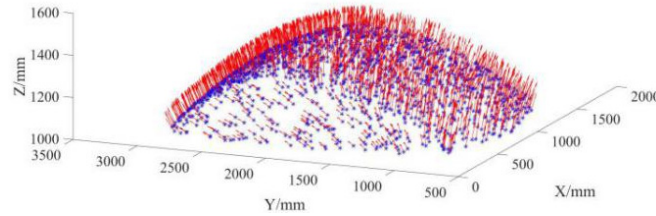


Figure 23. Normal vectors of target trajectory points.

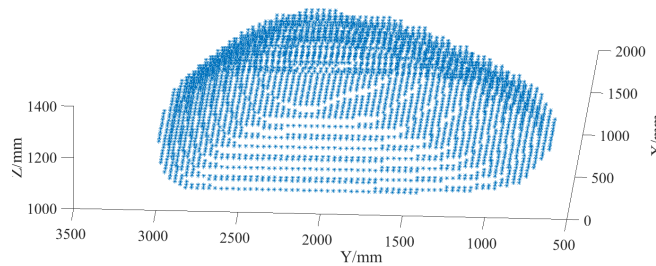


Figure 24. Compressed point cloud data.

6.2. Simulation of the Establishment of the Motion Platform Trajectory

After Equation (27) is given values T_{tool} and T_{dis} , the distribution of points at the ends of the UR5 robot joints is shown in Figure 25. By utilizing the smallest rectangle model in Figure 15, the smallest rectangle containing the target points was obtained, which is shown in Figure 26.

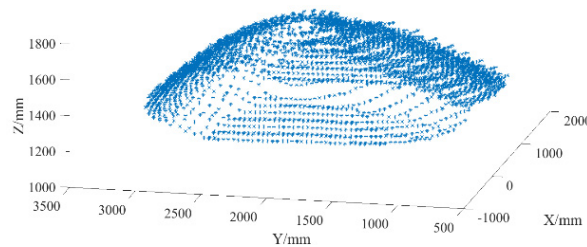


Figure 25. Distribution of points at the ends of the UR5 robot's joints.

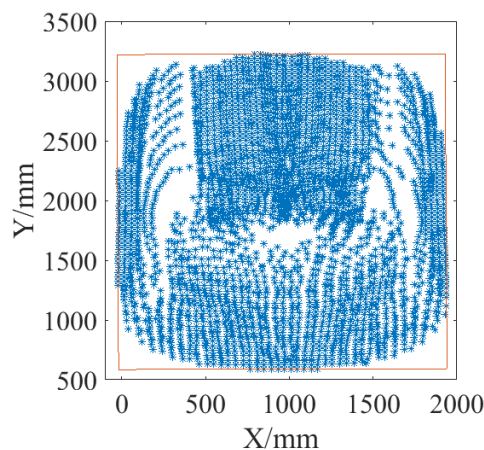


Figure 26. The minimum envelope rectangle.

The rectangle in Figure 26 has a width of 2635 mm and a length of 1959 mm. Data in Figure 26 were extracted and employed in Figure 16 and Equation (30) to obtain the preliminary partition model of the target points, which is shown in Figure 27.

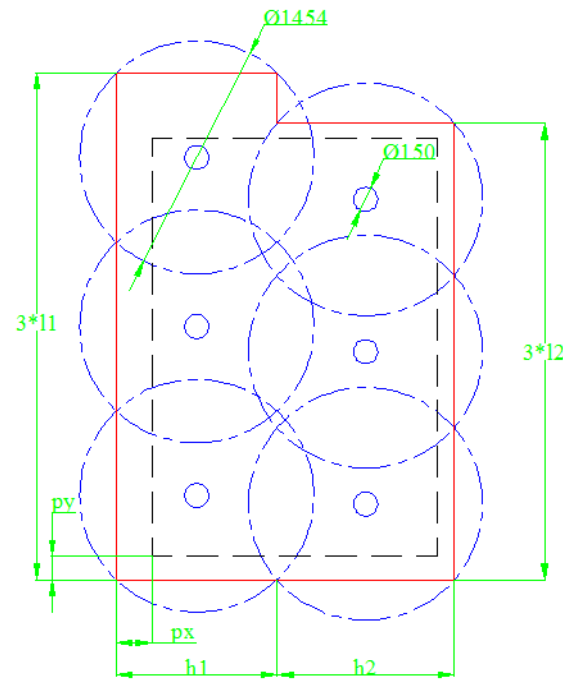


Figure 27. The preliminary partition model of the target points.

The model in Figure 27 was created by the exhaustive method. The result is shown in Figure 28, which describes the minimum number of target points in the non-recommended area when h_1 and h_2 are determined. The blue areas in the figure represent an invalid combination that does not satisfy Equation (30). In this optimal combination, h_1 equals 980 mm; h_2 equals 980 mm; P_x equals 7 mm; and P_y equals 3 mm.

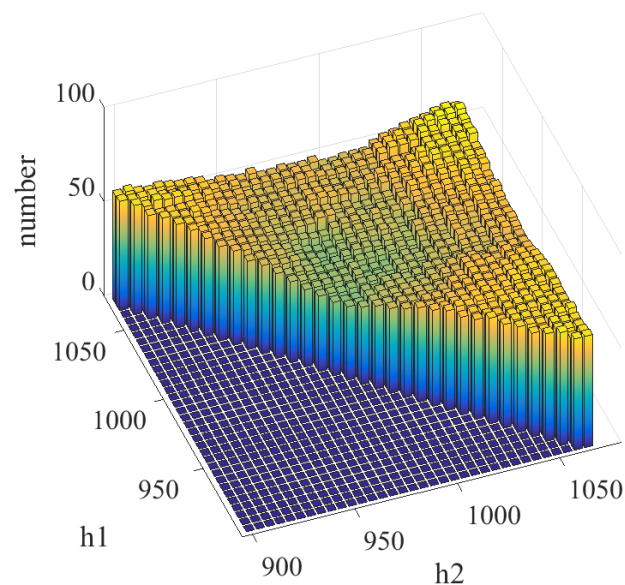


Figure 28. Statistical results of the objective function.

The model in Figure 27 was given values P_x , P_y , h_1 , and h_2 to obtain the partition model of the target points which are shown in Figure 29. The red circles in Figure 29 are the

maximum processing area and the non-recommended processing area of the UR5 robot. C_1 represents the coordinates of the i -th circle, and the coordinates of C_i are shown in Table 2.

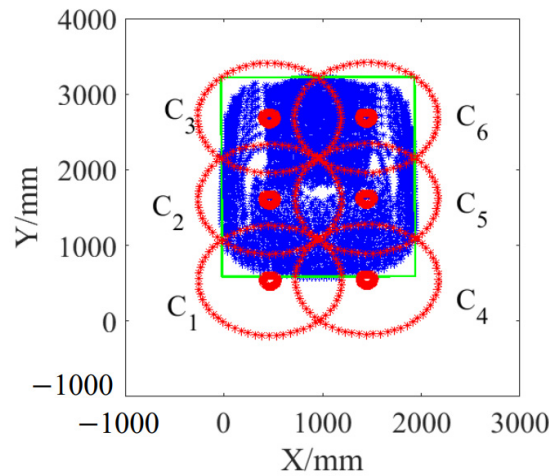


Figure 29. The partition model of the target points.

Table 2. The center point C_i .

C_i	X/mm	Y/mm
1	469	2685
2	465	1611
3	461	537
4	1451	552
5	1447	1622
6	1443	2691

All of the possibilities in Figure 29 were iterated, and the processing order with the smallest total movement of the motion platform was taken. The final painting order of the UR5 robot was $C_1 C_2 C_3 C_6 C_5 C_4$. The painting intervals of the UR5 robot are shown in Figure 30. The dots in different colors in Figure 30 represent different painting intervals.

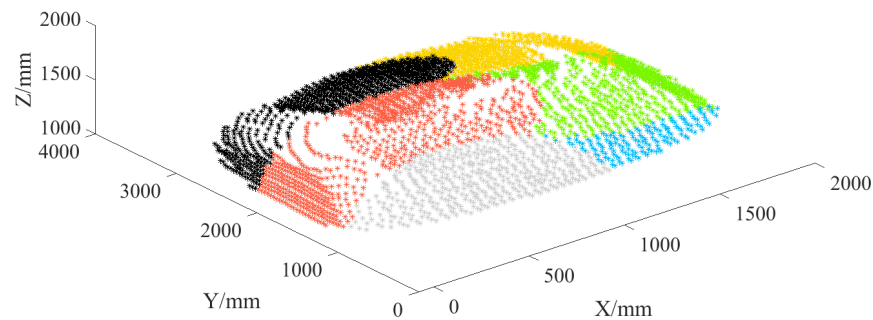


Figure 30. The painted zones of target points.

6.3. The Establishment of the Motion Platform Trajectory

The target points with the same color in Figure 30 were introduced into and utilized in GAs, and the population number was set to 5000. The maximum number of iterations is 25,000, and the cross probability was 0.9. K_1 equals 0.15, and K_2 equals 0.2 in the selection probability. The GA solution process is shown in Figure 31, and the final painting trajectory is shown in Figure 32.

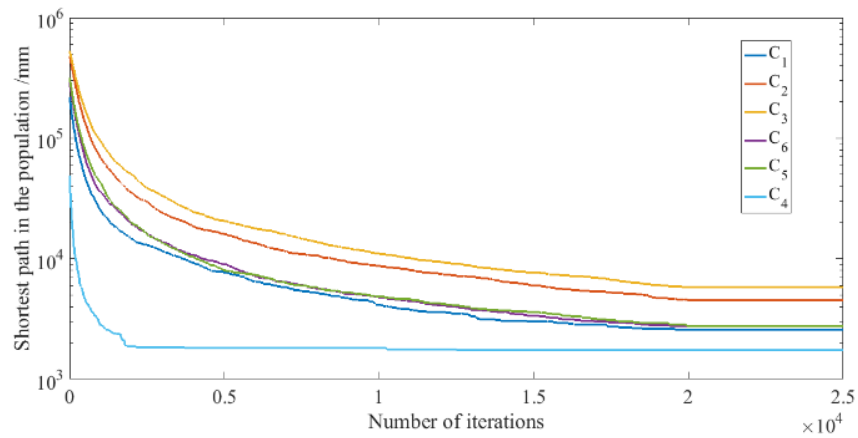


Figure 31. The iterative process of the genetic algorithm.

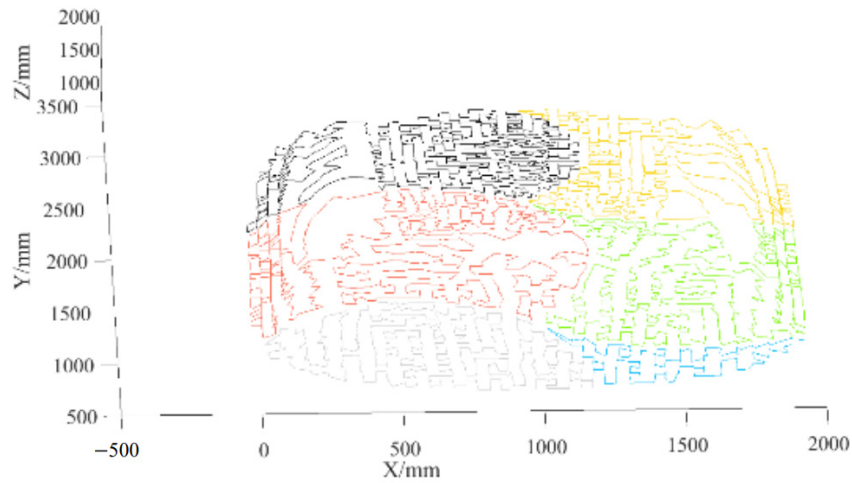


Figure 32. The final painting trajectory.

6.4. The Inverse Kinematics Simulation of the UR5 Robot

When the UR5 robot found its inverse kinematics, it generated multiple solutions. When the UR5 robot’s inverse kinematics were being solved, the angle values of the target points on the previous painting trajectory were harnessed as the initial angle values, and the target points on the final painting trajectory in each colored area with the smallest radian changes from the initial angle values were selected. The solution was the inverse kinematics of the target points. The position errors and axis errors of the statistical UR5 robot simulation are shown in Figures 33 and 34, respectively.

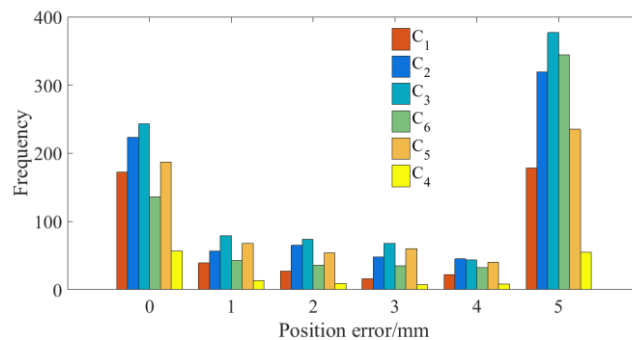


Figure 33. The UR5 robot’s position errors.

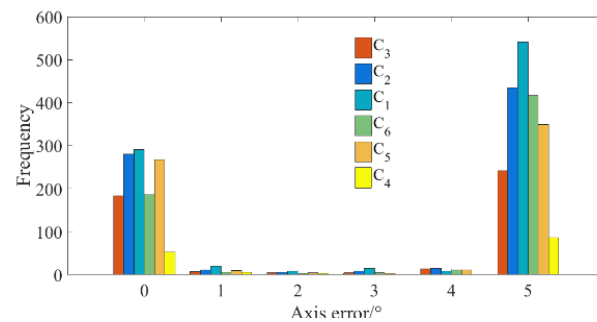


Figure 34. The UR5 robot's axis errors.

As per Figures 33 and 34, the errors are within the technical requirements of painting large objects, and the algorithm has satisfactory engineering practicability. Since the main solution in this paper was an analytical solution, when the joint radian of the UR5 robotic manipulator's motion platform in the recommended area needed to be solved, an iterative method was harnessed to solve Equations (14) to (18).

7. Discussion and Conclusions

This paper proposes an algorithm that employs a nine-axis robotic manipulator to automatically paint large objects. With the proposed algorithm, automatic processing of complex objects is achieved. The algorithm is divided into three aspects that consist of extraction of the target model, the establishment of the inverse kinematics model of the manipulator, and the planning of the target trajectory.

In the section of the proposed algorithm for extracting the target model, as per the STL file of the target trajectory, the triangles of the target trajectory are extracted. A full 3D data point cloud model is obtained with a triangle-based filling algorithm. Subsequently, the PCA algorithm is harnessed to identify the normal vectors of the target points inside the point cloud model. With the normal vectors, Rodrigues' rotation formula is harnessed to extract the pose of each point of the painting trajectory. Finally, the number of target points is compressed to reduce the time and space complexity of the algorithm so that the painting process requirements can be satisfied.

In the section of the proposed algorithm where the inverse kinematics model of the manipulator is established, in conformity with the processing range of the robot, the inverse solution algorithm is divided into the inverse solution of the recommended region and the inverse solution of the non-recommended region. The corresponding inverse kinematics model is established by employing the closed-form solution method and SQP, respectively.

During the planning of the painting trajectory, in line with the point cloud model of the target points, the minimum envelope rectangles of the target points are found. The target points are divided into different painting areas in accordance with the minimum rectangles. For each painting area, a TSP trajectory planning model based on GA is harnessed to plan the robot's painting trajectory with which the inverse kinematics model of the UR5 robot is created.

Not only does the trajectory created by the proposed algorithm consider the singularities of the kinematic joints of the UR5 robot joints, but it also helps the UR5 robot to paint large objects with precision and efficiency. Multiple reliable simulations of the painting process on the car roof surface were conducted, and the results show that the algorithm can meet the technical requirements of painting and that the algorithm has promising practicability.

The proposed algorithm has several benefits. The trajectory created by the proposed algorithm allows the motion platform of the UR5 robotic manipulator to have the least amount of movement while the UR5 robot paints a large object. Therefore, the proficiency of the painting process can be significantly improved. The standard triangle language (STL) file, algorithm of principal component analyses (PCA), and k-dimensional tree (k-d tree) were employed to obtain the point cloud model of the car roof to be painted. The

point cloud model was later converted into multiple traveling salesman problem (TSP) models, each of which was created with genetic algorithms (GAs). In this way, the UR5 robot could identify the object to be painted and generate a trajectory to paint the large object more efficiently.

However, limitations still exist in the current research. The closed-form solution method and SQP were employed to establish the inverse kinematics model of the UR5 robot. SQP is not intelligent enough and still requires a myriad of training solutions. If a more intelligent neural network model had been utilized, the preparation time for the entire painting process would have been reduced. In this research, only the painting trajectory was created, but the dynamics and velocity-planning of the UR5 robot were not involved. Therefore, future studies need to focus on a more intelligent neural network model that helps to establish the inverse kinematics model of the UR5 robot as well as the dynamics and velocity-planning of the UR5 robot.

Author Contributions: Conceptualization, F.L.; Data curation, J.W.; Formal analysis, M.Y. and K.Z.; Funding acquisition, J.W. and Q.W.; Investigation, M.Y. and F.L.; Methodology, M.Y., F.L. and K.Z.; Project administration, Q.W.; Resources, K.Z.; Software, F.L.; Supervision, J.W.; Visualization, K.Z.; Writing—original draft, K.F.; Writing—review & editing, M.Y. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The study did not report any data.

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

References

1. Craig, J.J. *Introduction to UR5 Robotics: Mechanics and Control*, 3rd ed.; Prentice Hall: Upper Saddle River, NJ, USA, 2008.
2. Goetz, J.; Kiesler, S.; Powers, A. Matching UR5 Robot Appearance and Behavior to Tasks to Improve Human-UR5 Robot Cooperation. In Proceedings of the 12th IEEE International Workshop on UR5 Robot and Human Interactive Communication, Millbrae, CA, USA, 2 November 2003; pp. 55–60.
3. Brogårdh, T. Present and future UR5 robot control development—An industrial perspective. *Annu. Rev. Control* **2007**, *31*, 69–79. [[CrossRef](#)]
4. Wang, Y.; Liu, S.; Xu, D.; Zhao, Y.; Shao, H.; Gao, X. Development and Application of Wall-Climbing UR5 robots. In Proceedings of the 1999 IEEE International Conference on UR5 Robotics and Automation (Cat. No.99CH36288C), Detroit, MI, USA, 10–15 May 1999; Volume 2, pp. 1207–1212.
5. Form, P.J.; Gravdahl, J.T.; Pettersen, K.Y. Kinematics of vehicle-manipulator systems. In *Vehicle-Manipulator Systems. Advances in Industrial Control*; Springer: London, UK, 2014; pp. 169–189. [[CrossRef](#)]
6. Ren, S.; Yang, X.; Xu, J.; Wang, G.; Xie, Y.; Chen, K. Determination of the base position and working area for mobile manipulators. *Assem. Autom.* **2016**, *36*, 80–88. [[CrossRef](#)]
7. Li, Z.; Zhao, D.; Zhao, J. Structure synthesis and workspace analysis of a telescopic painting UR5 robot. *Mech. Mach. Theory* **2019**, *133*, 295–310. [[CrossRef](#)]
8. Bureerat, S.; Pholdee, N.; Radpukdee, T.; Jaroenapibal, P. Self-adaptive MRPBIL -DE for 6D UR5 robot multiobjective trajectory planning. *Expert Syst. Appl.* **2019**, *136*, 133–144. [[CrossRef](#)]
9. Yin, S.; Ji, W.; Wang, L. A machine learning based energy efficient trajectory planning approach for industrial UR5 robots. *Procedia CIRP* **2019**, *81*, 429–434. [[CrossRef](#)]
10. Serralheiro, W.; Maruyama, N.; Saggin, F. Self-Tuning Time-Energy Optimization for the Trajectory Planning of a Wheeled Mobile UR5 robot. *J. Intell. Robot. Syst.* **2019**, *95*, 987–997. [[CrossRef](#)]
11. Kazim, I.J.; Tan, Y.; Li, R. Comparison study of the PSO and SBPSO on universal robot trajectory planning. *Appl. Sci.* **2022**, *12*, 1518. [[CrossRef](#)]
12. Kazim, I.J.; Tan, Y.; Qaseer, L. Integration of DE algorithm with PDC-APF for enhancement of contour path planning of a universal robot. *Appl. Sci.* **2021**, *11*, 6532. [[CrossRef](#)]
13. Al-shanoon, A.; Lang, H. Robotic manipulation based on 3-D visual servoing and deep neural networks. *Robot. Auton. Syst.* **2022**, *152*, 104041. [[CrossRef](#)]
14. Balanji, H.M.; Turgut, A.E.; Tunc, L.T. A novel vision-based calibration framework for industrial robotic manipulators. *Robot. Comput. Manuf.* **2022**, *73*, 102248. [[CrossRef](#)]

15. Vivas, A.; Sabater, J.M. UR5 Robot Manipulation Using Matlab/Simulink and ROS. In Proceedings of the 2021 IEEE International Conference on Mechatronics and Automation, Takamatsu, Japan, 8–11 August 2021. [[CrossRef](#)]
16. Araki, R.; Mano, K.; Hirano, T.; Hirakawa, T.; Yamashita, T.; Fujiyoshi, H. Iterative Coarse-to-Fine 6D-Pose Estimation Using Back-Propagation. In Proceedings of the 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Prague, Czech Republic, 27 September–1 October 2021. [[CrossRef](#)]
17. Sunil, V.B.; Pande, S.S. Automatic recognition of features from freeform surface CAD models. *Comput. Aided Des.* **2008**, *40*, 502–517. [[CrossRef](#)]
18. Hallmann, M.; Goetz, S.; Schleich, B. Mapping of GD&T information and PMI between 3D product models in the STEP and STL format. *Comput. Aided Des.* **2019**, *115*, 293–306. [[CrossRef](#)]
19. Deun, K.V.; Thorrez, L.; Coccia, M.; Hasdemir, D.; Westerhuis, J.A.; Smilde, A.K.; Mechelen, I.V. Weighted sparse principal component analysis. *Chemom. Intell. Lab. Syst.* **2019**, *195*, 103875. [[CrossRef](#)]
20. Bro, R.; Kjeldahl, K.; Smilde, A.K.; Kiers, H.A.L. Cross-validation of component models: A critical look at current methods. *Anal. Bioanal. Chem.* **2008**, *390*, 1241–1251. [[CrossRef](#)]
21. Markiewicz, P.J.; Matthews, J.C.; Declerck, J.; Herholz, K. Verification of predicted robustness and accuracy of multivariate analysis. *NeuroImage* **2011**, *56*, 1382–1385. [[CrossRef](#)] [[PubMed](#)]
22. Hu, L.; Nooshabadi, S. High-dimensional image descriptor matching employing highly parallel KD-tree construction and approximate nearest neighbor search. *J. Parallel Distrib. Comput.* **2019**, *132*, 127–140. [[CrossRef](#)]
23. Liao, H.; Wu, W.; Fang, D. The reduced space Sequential Quadratic Programming (SQP) method for calculating the worst resonance response of nonlinear systems. *J. Sound Vib.* **2018**, *425*, 301–323. [[CrossRef](#)]
24. Lee, J.H.; Jung, Y.M.; Yuan, Y.-X.; Yun, S. A subspace SQP method for equality constrained optimization. *Comput. Optim. Appl.* **2019**, *74*, 177–194. [[CrossRef](#)]
25. Singh, D.; Dar, B.A.; Kim, D.S. KKT optimality conditions in interval valued multiobjective programming with generalized differentiable functions. *Eur. J. Oper. Res.* **2016**, *254*, 29–39. [[CrossRef](#)]
26. Zhou, Y.; Luo, Q.; Chen, H.; He, A.; Wu, J. A discrete invasive weed optimization algorithm for solving traveling salesman problem. *Neurocomputing* **2015**, *151*, 1227–1236. [[CrossRef](#)]
27. Ahmadi, E.; Goldengorin, B.; G. Süer, A.; Mosadegh, H. A hybrid method of 2-TSP and novel learning-based GA for job sequencing and tool switching problem. *Appl. Soft Comput.* **2018**, *65*, 214–229. [[CrossRef](#)]
28. Liu, F.; Zeng, G. Study of genetic algorithm with reinforcement learning to solve the TSP. *Expert Syst. Appl.* **2009**, *36*, 6995–7001. [[CrossRef](#)]