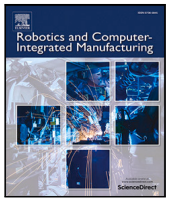
Robotics and Computer–Integrated Manufacturing 80 (2023) 102475

Contents lists available at ScienceDirect

Robotics and Computer-Integrated Manufacturing

journal homepage: www.elsevier.com/locate/rcim



Full length article

# An online collision-free trajectory generation algorithm for human–robot collaboration

Yanzhe Wang [a,b], Lai Wei [c], Kunpeng Du [d], Gongping Liu [d], Qian Yang [a,b], Yanding Wei [a,b,*], Qiang Fang [a,**]

[a] The State Key Laboratory of Fluid Power and Mechatronic Systems, School of Mechanical Engineering, Zhejiang University, Hangzhou 310027, China
[b] Key Laboratory of Advanced Manufacturing Technology of Zhejiang Province, School of Mechanical Engineering, Zhejiang University, Hangzhou 310027, China
[c] School of Data Science, The Chinese University of Hong Kong, Shenzhen, Shenzhen 518172, China
[d] AVIC Xi'an Aircraft Industry (Group) Company Ltd., Xi'an, Shaanxi 710089, China

ARTICLE INFO

*Keywords:*
Dynamic trajectory generation
Neural network
Online collision avoidance
Smooth trajectory
Human–robot collaboration

ABSTRACT

The premise of human–robot collaboration is that robots have adaptive trajectory planning strategies in hybrid work cell. The aim of this paper is to propose a new online collision avoidance trajectory planning algorithm for moderate dynamic environments to insure human safety when sharing collaborative tasks. The algorithm contains two parts: trajectory generation and local optimization. Firstly, based on empirical Dirichlet Process Gaussian Mixture Model (DPGMM) distribution learning, a neural network trajectory planner called Collaborative Waypoint Planning network (CWP-net) is proposed to generate all key waypoints required for dynamic obstacle avoidance in joint space according to environmental inputs. These points are used to generate quintic spline smooth motion trajectories with velocity and acceleration constraints. Secondly, we present an improved Stochastic Trajectory Optimization for Motion Planning (STOMP) algorithm which locally optimizes the generated trajectories of CWP-net by constraining the trajectory optimization range and direction through the DPGMM model. Simulations and real experiments from an industrial use case of human–robot collaboration in the field of aircraft assembly testing show that the proposed algorithm can smoothly adjust the nominal path online and effectively avoid collisions during the collaboration.

## 1. Introduction

In industry, robotic automation ensures efficient and repeatable production in manufacturing. However, traditional robots merely follow entirely stationary tasks and cannot cope with the uncertain changes in the environment. Human intervention is required to deal with this uncertainty and variability. With the new demands of the intelligent manufacturing in Industry 4.0, the use of industrial robots is gradually shifting from the traditional work that is isolated from humans to a collaborative work where they share the workspace with their human colleagues. Combined with the advantages such as cognitive abilities, flexibility from humans and agility, accuracy, and repeatability from robots, Human–Robot Collaboration (HRC) is becoming a popular topic in the robotics research field. Achieving the goal of developing safe collaborative robots will enlarge the scope for robotic applications and create greater value in the industrial field. The standard ISO 10218 and the technical specification TS 15066 define the safety requirements for collaborative robots [1]. In HRC, collision avoidance is specified as one of the most important factors in collaborative safety. Today, the majority of research on HRC is still limited to simulation cases. Collision avoidance modes in industrial applications are also mainly implemented by decelerating and braking near the obstacles. Therefore, research on intelligent collision avoidance strategies for collaborative robots in dynamic environments still needs to be explored.

Traditionally, offline programming is sufficient to make robots complete a fixed-path task. The most influential path planning algorithms are sampling-based path planning algorithms including the multi-query probabilistic roadmap (PRM) [2] and the single-query fast search random tree (RRT) [3] algorithms and their variants [4–6]. The sampling-based algorithms have efficient solving speed in low dimensional problems, but the computational complexity gradually increases as the dimensional increases. Meanwhile, the output feasible solution lacks the control of path quality, and requires subsequent smoothing optimization.

* Corresponding author at: The State Key Laboratory of Fluid Power and Mechatronic Systems, School of Mechanical Engineering, Zhejiang University, Hangzhou 310027, China.
** Corresponding author.
*E-mail addresses:* weiyd@zju.edu.cn (Y. Wei), fangqiang@zju.edu.cn (Q. Fang).

https://doi.org/10.1016/j.rcim.2022.102475
Received 19 February 2022; Received in revised form 5 August 2022; Accepted 11 October 2022

Unlike sampling-based methods, optimization-based methods obtain the optimal solutions under the given objective function by exploring the entire configuration space. The classical optimization-based methods for motion trajectory planning consists of two different branches: gradient-based methods and stochastic optimization methods. A well-known gradient-based method is Covariant Hamiltonian Optimization for Motion Planning (CHOMP) [7]. CHOMP uses a covariant gradient descent technique that minimizes the combined cost of trajectory smoothness and obstacle cost, enabling obstacle avoidance motion planning for a 6 degree-of-freedom (DOF) robotic arm. A representative of stochastic optimization methods is Stochastic Trajectory Optimization for Motion Planning (STOMP) [8], which updates the trajectory based on the stochastic gradient estimation generated by noise trajectories. Stochastic optimization methods overcome the limitation of local minima and improve the planning efficiency compared to gradient-based methods. With the continuous development of advanced numerical optimization algorithms, optimization-based algorithms have also improved significantly in terms of computational efficiency and global search for optimal solutions. Kim et al. [9] proposed a robot motion planner using particle swarm optimization (PSO) algorithm that satisfies multi-objective constraints, and tests on a 3-DOF robot showed good convergence efficiency. Xidias [10] computed time-optimal trajectories in 3D workspaces by Genetic Algorithm with multiple population. Recently, Chen et al. [11] solved an optimal trajectory planning problem with time, energy and jerk as multiple objectives based on an improved immune clone selection algorithm. The generated trajectories perform well, but the planner does not have obstacle avoidance capability due to the lack of obstacle constraints. In general, the optimization-based methods described above facilitate the generation of cost-saving and task-fitting ideal trajectories by defining appropriate cost functions and various types of constraints in motion. However, a major drawback of such methods is the high computational burden, which makes it challenging to apply to dynamic scenarios with variable initial conditions and complex task constraints.

For dynamic trajectory planning in HRC scenarios, various algorithms that improve human safety and planning efficiency have been developed recently. Due to the simple structure and good real-time performance, the artificial potential field (APF) method [12] becomes one of the commonly used real-time obstacle avoidance algorithms. Liu et al. [13] established APF from online estimated human hand velocities and determined collision avoidance directions based on the control law. Safeea et al. [14] proposed a repulsive vector shaping method for APF, capable of modifying the original nominal paths in collaborative tasks online to avoid collision. However, APF methods have the disadvantage of being susceptible to local minima regions and possible uncertainties in the newly generated trajectories when the repulsive and gravitational forces are similar.

There are also studies that improve optimization-based methods for dynamic environments. Incremental Trajectory Optimization for Motion Planning (ITOMP) [15], which alternates planning and execution steps based on STOMP, has the capability of real-time motion planning. Oguz et al. [16] proposed a stochastic optimization framework supported by updated parameterization and human motion prediction for HRC trajectory planning. The above single-step incremental trajectory generation method has good dynamic performance, but the speed and acceleration trajectory quality during continuous motion need to be optimized due to the absence of a forward-looking trajectory smoothing method. In contrast, Kim et al. [17] proposed an online near time-optimal trajectory planning method based on path constrained time-optimal motion. The dynamic trajectory generation time is close to optimal and the trajectory transitions are smooth. However, the authors focus on the generation of continuous trajectories, and the way to obtain the key path points for obstacle avoidance specified by higher-order commands is not given. Similarly, polynomial functions proposed by [18] for online collision avoidance trajectory also has smooth transitions.

Recently, more and more researches have focused on introducing machine learning techniques into HRC for improving dynamic performance and adapting to complex tasks. Mainprice et al. [19] used inverse optimal control to learn sample trajectories of human motion, and the prediction of human actions were used for subsequent STOMP-based trajectory planning. The stochastic optimization process of Oguz et al. [16] is also based on a Probabilistic Movement Primitive (ProMP) model for human motion prediction implemented in the offline training. Such methods plan robot trajectories based on model prediction results by learning human motion guidelines. However, this strategy is somewhat influenced by the dynamic habits of human collaborators and the degree of response to robot motion. In contrast, another learning-based scheme is to learn the robot's obstacle avoidance strategies under different obstacles, with no motion criterion requirement for humans, such as [20]. This way of adapting the robot to the human can make the collaborator more comfortable and natural.

Due to the low time complexity and superior nonlinear fitting performance, neural network-based motion planning research is a recent hot direction in this field. Meziane et al. [20] used supervised learning of a two-layer neural network to generate path points for dynamic obstacle avoidance of collaborative robots. Their algorithm plans in Cartesian space, which is only applicable to motion planning of end-effectors and does not guarantee that the robot links do not create collision with humans. Qureshi et al. [21] proposed a motion planning network (MPnet) that inputs environmental information and generates single-step path points connecting the initial and target configurations of the robot. Bency et al. [22] used a recurrent neural network to output configuration space path points step-by-step, but the network has no input of environmental information. Ichter et al. [23] proposed a method for learning the sampling distribution of a Sampling-based motion planner (SBP) using a conditional variational autoencoder. Duguleana et al. [24] developed a dual neural network based obstacle avoidance network that uses Q-learning to iteratively update the neural network weights. However, there is a reset process for the network weights, which is risky when performing dynamic tasks. The network structure of the above methods determines that only a single path point or discrete informed samples can be generated in each cycle of path planning. The motion pattern of the robot is single step incremental. The path cost and trajectory smoothness of such methods in dynamic environments cannot be planned uniformly.

To the best of the authors' knowledge, our study differs from existing studies in the following ways.

(*i*) In this paper, we propose CWP-net, a neural network joint trajectory planner based on empirical DPGMM distribution. Unlike common neural network algorithms with single-step incremental output, CWP-net is able to generate all subsequent critical obstacle avoidance path points in joint space in a single step based on environmental information. Simultaneously, the entire motion trajectory is optimized based on joint angle, angular velocity and angular acceleration constraints. While ensuring the continuity of velocity and acceleration, it reduces the impact of acceleration on the robot joints and improves the smoothness of the trajectory.

(*ii*) This paper proposes an improved STOMP trajectory optimization algorithm for the local optimization of the trajectory generated by CWP-net. The algorithm is based on the solved empirical DPGMM distribution parameters constraining the stochastic optimization range and direction of joint trajectories. It converges to the ideal optimization range faster than the STOMP algorithm using fixed range stochastic exploration, further improving the efficiency of the algorithm.

(*iii*) Overall, the safe trajectory generation algorithm proposed in this paper combines the offline path of the robot with online collision avoidance trajectory replanning. It can modify the default task path online based on dynamic obstacle information to generate smooth collision-free trajectories, which is suitable for similar industrial collaboration scenarios.

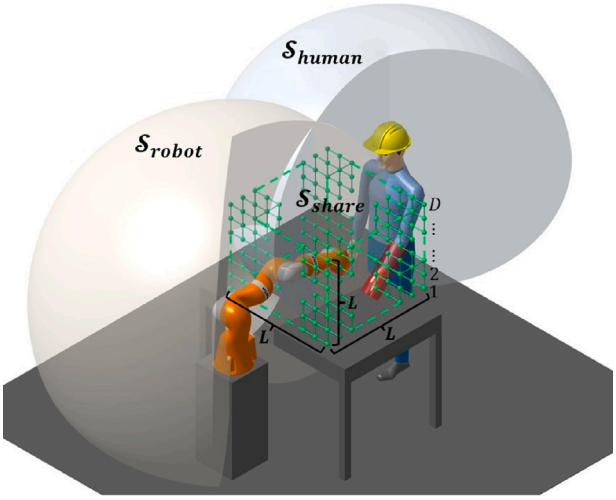



**Fig. 1.** Representation of HRC scenarios.

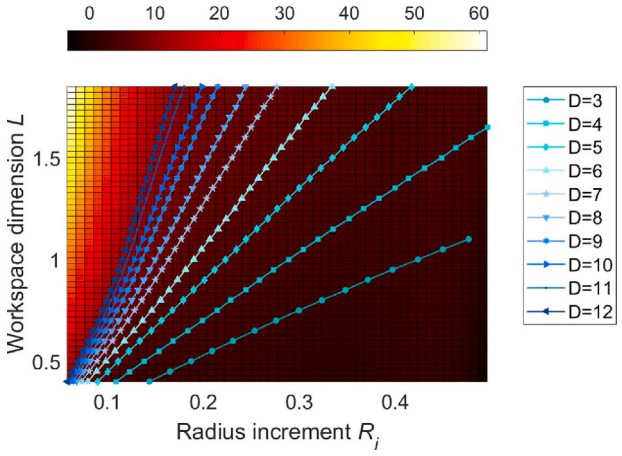



**Fig. 2.** The required reference discrete resolution $D$ according to the edge length of the shared workspace $L$ and the error increment of the modeling radius to be limited $R_i$. (The colormap is obtained by interpolating the blue scattered data in the figure). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

More specifically, the CWP-net-based planner has an excellent real-time performance and is suitable for online trajectory generation in dynamic environments. However, currently our improved STOMP designed for ensuring the completeness and safety of the online trajectory planning framework is only applicable to moderate low speed HRC scenarios. Computational efficiency improvements will be made in the future with reference to [15,16].

The rest of the paper is organized as follows. In Section 2, the HRC scenario of the proposed approach is described, modeling the workspace for collaborative tasks. Section 3 introduces the proposed deep neural network motion planner CWP-net, and the process of solving the DPGMM distribution model for the training data. Section 4 presents the online trajectory planning algorithm proposed in this paper and proposes an improved STOMP motion optimization algorithm. Then, in Section 5, simulations and real experiments on obstacle avoidance during HRC of an industrial use case are designed, and the performance of the method in this paper is discussed and analyzed. Finally, Section 6 concludes the paper.

## 2. HRC scenario modeling

The method proposed in this paper is applicable to solve the dynamic collision avoidance problem similar to HRC scenario illustrated in Fig. 1. The robot workspace is $\mathcal{S}_{robot} \subset \mathbb{R}^3$ and the human workspace is $\mathcal{S}_{human} \subset \mathbb{R}^3$. The shared workspace $\mathcal{S}_{share} \subset \mathbb{R}^3$ is a subset of the intersection of the robot workspace and the human workspace. In the shared workspace, the robot needs to plan its task motion trajectory online while not interfering with humans. $\mathcal{S}_{share}$ is determined according to the work task and is defined as a cubic space with dimensions $L \times L \times L$. The human arm in the collaboration is modeled as an envelope cylinder with a radius of 0.06 m. The arm position and pose are represented by the envelope geometry features. Arm localization can be achieved by sensors such as cameras, IMUs, lasers and magnetic trackers [14,20].

Since human motion is dynamic and uncertain, collision-free path planning must be done in real time. Therefore, we adopt a discretization space approach [20,25] to characterize the state of the HRC environment, as shown in Fig. 1. In the collaboration process, $\mathcal{S}_{share}$ contains the obstacle space $\mathcal{X}_{obs}$ formed by the human arm, and the robot free movable space $\mathcal{X}_{free} = \mathcal{S}_{share} - \mathcal{X}_{obs}$. The discrete resolution of the workspace determines the characterization accuracy of $\mathcal{X}_{obs}$. The larger the shared workspace dimensions, the higher the discrete resolution required to achieve the specified obstacle characterization accuracy. Typically, the resolution $D \times D \times D$ for the workspace is usually manually set by the researchers, so there is no existing standard to refer to. In this paper, we propose a uniform sampling-based empirical formula to determine the proper discrete resolution $D$, as shown in Fig. 2. This empirical formula (Eq. (1)) measures the reference discrete resolution $D$ under the given parameters, i.e., the edge length of the shared workspace $L$ and the expected increment of the modeled cylindrical size of the arm envelope over the actual radius $R_i$. More details can be found in Appendix A.

**Table 1**
The polynomial parameters of Eq. (1).

| $p_{00}$ | $p_{01}$ | $p_{10}$ | $p_{11}$ | $p_{20}$ | $p_{21}$ |
|---|---|---|---|---|---|
| 8.235 | 74.7 | −400.1 | −943.6 | 4705 | 4707 |
| $p_{30}$ | $p_{31}$ | $p_{40}$ | $p_{41}$ | $p_{50}$ | |
| −22580 | −10350 | 48190 | 8355 | −37910 | |

$$D\left(R_i, L\right) = \left[1, R_i, R_i^2, R_i^3, R_i^4, R_i^5\right] \begin{bmatrix} p_{00} + p_{01} L \\ p_{10} + p_{11} L \\ p_{20} + p_{21} L \\ p_{30} + p_{31} L \\ p_{40} + p_{41} L \\ p_{50} \end{bmatrix} \tag{1}$$

where $R_i$ is the increment of cylindrical envelope radius over the actual radius. $L$ is the edge length of the shared workspace. $D(R_i, L)$ is the value of required discrete resolution under given $R_i$ and $L$. The polynomial parameters $p$ can be found in Table 1. Taking $R_i$ of 0.12 m and $L$ of 0.55 m as an example, $D(R_i, L)$ can be calculated as 5.2092 by Eq. (1). Therefore, we choose $D = 5$ and set the resolution as $5 \times 5 \times 5$.

Since the value of spatial discrete resolution directly affects the structure of our proposed neural network below, the selection of resolution requires a trade-off between collision avoidance accuracy, computation time, and neural network complexity. Therefore, the number of discrete in the range of discrete reference lines should be chosen as small as possible under the condition that the specified collision avoidance distance is satisfied. In addition, the empirical Eq. (1) is only for fitting the distribution near the sample data, and is not applicable beyond the range of Fig. 2, or if $R_i$ is chosen too small.

According to the discrete value D determined by Eq. (1), $\mathcal{S}_{share}$ is discretized into $D \times D \times D$ subspaces. For each subspace, the subspace occupied by obstacles is represented by the value "1", and the rest of the subspaces are represented by "0". After flattening the three-dimensional space into a one-dimensional vector, the state of the shared

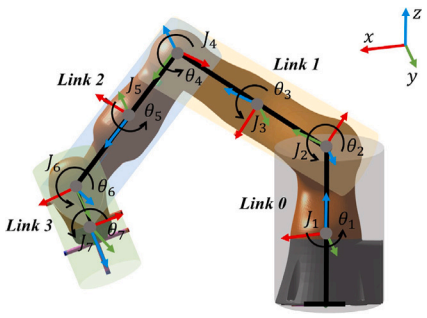



**Fig. 3.** A 7-DOF articulated manipulator and its link model.

workspace at any moment can be represented by a one-dimensional vector of $D^3$ elements, $\boldsymbol{I}_{obs} = \left[i_1, i_2, \ldots, i_{D^3}\right]$.

## 3. DPGMM-based CWP-net

### 3.1. Off-line trajectory experience learning

In this section, we propose a modeling method for the point distribution of HRC joint obstacle avoidance paths based on the DPGMM [26]. By adaptive learning of historical task paths and obstacle avoidance paths generated from simulations of different obstacle distribution, starting configurations and target configurations, information on the joint angle distribution of different obstacle avoidance path is extracted. This distribution model is used as the training label for the deep neural network path planner in the later part of the paper.

Our HRC system uses the KUKA LBR iiwa 14 robot, a 7-DOF redundant collaborative robot. As shown in Fig. 3, the robot's obstacle avoidance action is determined only by the postures of Link 1, 2, and 3. Therefore, the seventh rotational joint at the end of the kinematic chain is considered as a redundant joint. We only consider the angles of the 1st to 6th joints in the learning part. As for the motion planning for 7th joint, there is no need to constrain the intermediate waypoints, which can be directly interpolated by the initial and target configurations. Specifically, according to the decisive joints that control the orientation of each link, three DPGMMs are used to model the distributions of $(\theta_1,\ \theta_2)$, $(\theta_3,\ \theta_4)$ and $(\theta_5,\ \theta_6)$, respectively, and encode the configuration space of the empirical path into a finite number of known Gaussian parameters.

The nonparametric DPGMM [26], also known as infinite Gaussian Mixture Model, is an extended form of the finite Gaussian Mixture Model when the number of submodels tends to infinity. It can calculate the number of effective submodels adaptively based on the data, thus avoiding the problem of insufficient model accuracy or overfitting. The DPGMM distribution model building process for obstacle avoidance trajectory experience is expressed as

$$\begin{aligned} \boldsymbol{\pi} \left| \alpha \right. &\sim Dir(\alpha) \\ \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k \left| \boldsymbol{\mu}_0, \boldsymbol{\Sigma}_0, \kappa_0, \upsilon_0 \right. &\sim NW\left(\boldsymbol{\mu}_0, \boldsymbol{\Sigma}_0, \kappa_0, \upsilon_0\right) \\ z_i \left| \boldsymbol{\pi} \right. &\sim Mult(\boldsymbol{\pi}) \\ x_i \left| z_i, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k \right. &\sim \mathcal{N}\left(\boldsymbol{\mu}_{z_i}, \boldsymbol{\Sigma}_{z_i}\right) \end{aligned} \tag{2}$$

where $\alpha$ is the concentration parameter of the model. The Dirichlet prior distribution of the probability parameter $\pi$ is obtained from $Dir(\alpha)$, which can be constructed using $GEM(\alpha)$ (stick-breaking). The prior distribution of the submodel mean parameter $\boldsymbol{\mu}_k$ and the covariance parameter $\boldsymbol{\Sigma}_k$ is the Gaussian-Wishart distribution with parameters $\boldsymbol{\mu}_0, \boldsymbol{\Sigma}_0, \kappa_0, \upsilon_0$, denoted by $NW\left(\boldsymbol{\mu}_0, \boldsymbol{\Sigma}_0, \kappa_0, \upsilon_0\right)$. $Mult(\boldsymbol{\pi})$ indicates that the random indicator variable $z_i$ obeys a multinomial distribution with probability parameter $\boldsymbol{\pi}$. The final representation by $z_i = k$ indicates that the $i$th data $x_i$ belongs to the $k$th Gaussian mixture submodel.

The robot configuration space (C-space) is denoted as $\mathcal{C} \subset \mathbb{R}^d$ ($d$ is the C-space dimension), which contains the obstacle space $\mathcal{C}_{obs}$ and $\mathcal{C}_{free} = \mathcal{C} - \mathcal{C}_{obs}$. Given an obstacle $\mathcal{X}_{obs}$, a robot collaborative task starting configuration $\boldsymbol{c}_0$ and a target configuration $\boldsymbol{c}_t$, a discrete path sequence $\boldsymbol{\xi} = \left[\boldsymbol{c}_0, \boldsymbol{c}_1, \ldots, \boldsymbol{c}_t\right]$ is obtained by planning in $\mathcal{C}_{free}$. The path is capable of achieving task path connectivity under the condition of complete obstacle avoidance. The number of discrete path points $t$ needs to be chosen according to the length and complexity of the collision-free path, and the Douglas–Peucker algorithm [27] is used to compress the number of discrete trajectory points of each joint in C-space, and try to choose a smaller $t$ under the condition of conforming to the trajectory morphology. According to different state configurations, $N$ sample path data $\boldsymbol{\xi}_i = \left[\boldsymbol{c}_0^i, \boldsymbol{c}_1^i, \ldots, \boldsymbol{c}_t^i\right]$ for $\mathcal{X}_{obs}^i, \boldsymbol{c}_0^i, \boldsymbol{c}_t^i, i = 1, 2, \ldots, N$ can be obtained.

According to the spatial postures of robot links, path data in C-space are dimensionally reduced and split into three parts, $\boldsymbol{\psi}_1 = [(\theta_1,\ \theta_2)_0, (\theta_1,\ \theta_2)_1, \ldots, (\theta_1,\ \theta_2)_t]$, $\boldsymbol{\psi}_2 = [(\theta_3,\ \theta_4)_0, (\theta_3,\ \theta_4)_1, \ldots, (\theta_3,\ \theta_4)_t]$, and $\boldsymbol{\psi}_3 = [(\theta_5,\ \theta_6)_0, (\theta_5,\ \theta_6)_1, \ldots, (\theta_5,\ \theta_6)_t]$. Based on $\boldsymbol{\psi}_1$, $\boldsymbol{\psi}_2$ and $\boldsymbol{\psi}_3$, the spatial pose distribution models of the sample paths of links 1, 2 and 3 are modeled by Eq. (2), respectively. The EM algorithm [28] is used to solve the DPGMM model parameters, and the obtained distribution model $\Lambda_{k_m}^m$ is expressed as

$$\begin{aligned} \left\{ \Lambda_{k_m}^m \ : \boldsymbol{\mu}_{k_m}, \boldsymbol{\Sigma}_{k_m} \right\} &= DPGMM\left(\boldsymbol{\psi}_m\right), \\ k_m = 1, 2, \ldots, N_{k_m}; m &= 1, 2, 3 \end{aligned} \tag{3}$$

where $m$ is denoted as the link label, $k_m$ denotes the $k_m$ submodel of link $m$, and the total number of submodels is $N_{k_m}$. $\boldsymbol{\mu}_{k_m}$ and $\boldsymbol{\Sigma}_{k_m}$ are the submodel Gaussian distribution parameters.

Based on the distribution model, any combination of these three link attitudes can form a specific robot configuration, denoted as $\left\{\Lambda_i^m, \Lambda_j^m, \Lambda_k^m\right\}, i = 1, 2, \ldots, N_{k_1}, j = 1, 2, \ldots, N_{k_2}, k = 1, 2, \ldots, N_{k_3}$. Therefore, a total of $N_{k_1} \times N_{k_2} \times N_3$ different robot configurations can be created, as shown in the Encoding section in Fig. 4. The sample discrete path sequence can be expressed as $\boldsymbol{\xi}_i = \left[\boldsymbol{c}_0^i, \boldsymbol{c}_1^i, \ldots, \boldsymbol{c}_t^i\right] = \left[l_0^i, l_1^i, \ldots, l_t^i\right], i = 1, 2, \ldots, N$, where $l_t^i$ denotes the configuration label of the path point at moment $t$ of the $ith$ path. To facilitate subsequent network processing, the labels of all paths are formed into $t + 1$ sets $\boldsymbol{L}_t$ based on the moment $t$ to which they belong. Each $\boldsymbol{L}_t$ is reordered and normalized according to its own number. Thus, a normalized label sequence for each sample path is obtained as $\bar{\boldsymbol{\xi}}_i = \left[\bar{l}_0^i, \bar{l}_1^i, \ldots, \bar{l}_t^i\right], i = 1, 2, \ldots, N$.

### 3.2. Neural network model and training

In this section, a feedforward deep neural network is proposed, called Collaborative Waypoint Planning network (CWP-net) . CWP-net is able to learn the distribution information of the DPGMM model established in Section 3.1, and output all the key discrete collision-free waypoints of the subsequent planned path with the initial configuration, target configuration, and environmental obstacle information as inputs. The model building and learning process is as follows.

In the designed simulation, training data (containing discrete path trajectories and environmental obstacle configuration information) are divided into an input part and an output part for training the neural network. The form of the input part and output part is shown in the CWP-net part of Fig. 4. The input of CWP-net is a one-dimensional vector $\boldsymbol{x}_{in} = \left\{c_0, c_t, \left[\boldsymbol{I}_{obs}\right]\right\}$, containing $D^3$+2 elements, where the values of $c_0$ and $c_t$ are the corresponding normalized configuration labels $\bar{l}_0$ and $\bar{l}_t$, $\boldsymbol{I}_{obs} = \left[i_1, i_2, \ldots, i_{D^3}\right]$. The outputs of the CWP-net are the critical waypoints in each discrete path trajectory as $\boldsymbol{y}_{out} = \left\{c_1, c_2, \ldots, c_T\right\}$, where $T$ is determined by the maximum number of discrete trajectory points in the total training data. Depending on the number of critical waypoints for each data, successive elements starting from $c_1$ are assigned the corresponding normalized configuration labels, and subsequent empty elements are assigned a value of "−1". For example, the output data in Fig. 4, with the number of critical path points being

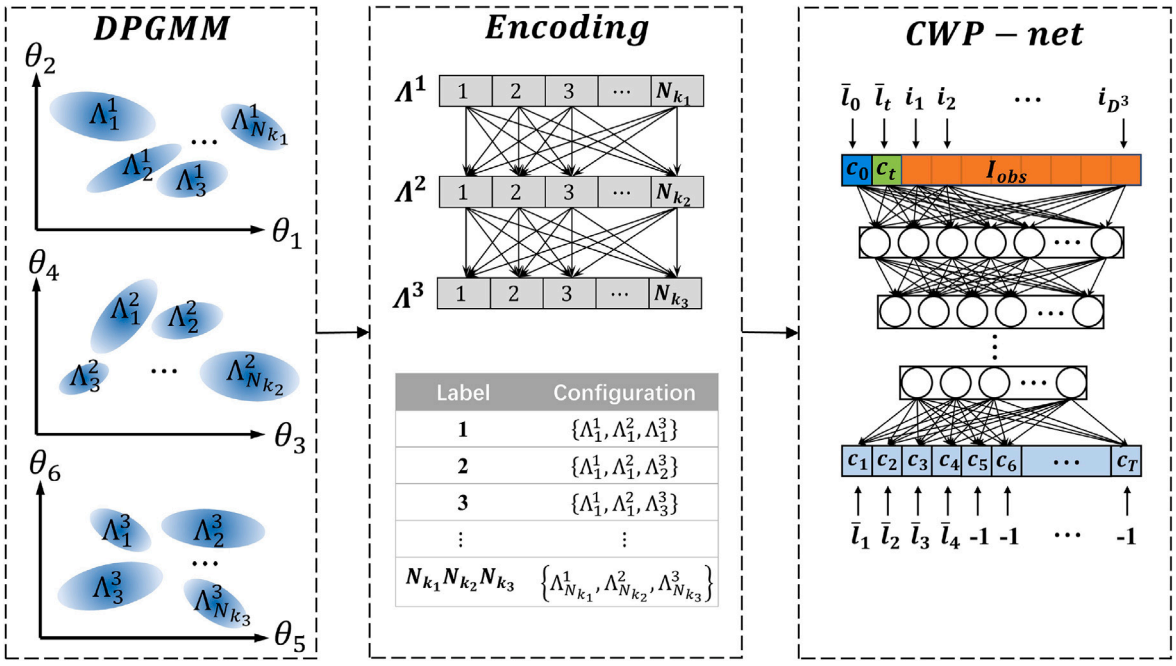



**Fig. 4.** Schematic diagram of the training process and network structure of the proposed neural network motion planner CWP-net.

4, corresponds to the sequential sequence label $\left[\bar{l}_1, \bar{l}_2, \bar{l}_3, \bar{l}_4\right]$, and the last $T-4$ bits of the output sequence are filled with "−1".

Referring to the guidelines for rough estimation of hidden neuron structure provided by Shibata et al. [29], the number of hidden layer neurons in CWP-net can be initially determined by

$$\begin{cases} H_{m+1} = H_m^2 / H_{m-1} & (1 \le m \le ht-2) \\ H_0 = D^3 + 2, \quad H_1 = 2H_0/3, \quad H_{ht-1} > H_{ht} \end{cases} \tag{4}$$

where $H_0$ is the number of neurons in the input layer, $D$ is the shared workspace discrete resolution, and $H_{ht}$ is the number of neurons in the output layer. The number of hidden layers and the number of neurons in each layer of CWP-net are determined by the recursive formula of Eq. (4). It should be noted that the computational complexity increases as the volume of the network increases. The pruning of the network structure should be further weighed based on factors such as workspace discrete resolution, path complexity, and training data.

The training process of the network uses a back propagation algorithm based on gradient descent. If the output vector is different from the desired target, it indicates that the computational result is wrong. The computed error is back propagated to the hidden layer, and the response error is continuously minimized by adjusting the network weight parameters in the training phase. The training objective function is the Mean Square Error (MSE) loss between the predicted path label and the target path label, as shown in Eq. (5).

$$Loss = \frac{1}{N} \sum_{i}^{M} \sum_{j=1}^{T} \left\| \hat{c}_{i,j} - c_{i,j} \right\|^2 \tag{5}$$

where $\hat{c}_{i,j}$ is the normalized path label predicted by CWP-net and $c_{i,j}$ is the normalized target path label. $M$ is the number of paths used for training. $N$ is the averaging term.

## 4. Online trajectory planning

In this section, an online collision avoidance trajectory planning algorithm for dynamic environments is proposed for ensuring HRC safety. First, the implementation and principles of the algorithm are given in Section 4.1. Then, for the optimization method used in the algorithm, an improved STOMP motion optimization algorithm, referred to as SWOA, is proposed in Section 4.2 for locally optimizing trajectories.

### 4.1. Trajectory planning algorithm

**Algorithm 1:** Online trajectory planning

**input :** ∘ $c_0$ – Start configuration;
∘ $c_t$ – Goal configuration;
∘ $I_{obs}$ – Obstacle Information;
**output:** ∘ $\phi$ or $\phi_{replan}$ – Collision-free trajectory.

$I_{obs} \leftarrow$ *Update the workspace state*;
$\xi \leftarrow$ CWPnet$(c_0, c_t, I_{obs})$;
$\phi \leftarrow$ Interpolation$(\xi)$;
**if** IsCF$(\phi, I_{obs})$ **then**
  // IsCF - IsCollisionFree;
  **Return** $\phi$
**else**
  **for** $i \leftarrow 0$ **to** size$(\phi)-1$ **do**
    **if** IsCF$(\left[\phi_i, \phi_{i+1}\right], I_{obs})$ **then**
      $\phi_{replan} \leftarrow \phi_{replan} \cup \left[\phi_i, \phi_{i+1}\right]$;
    **else**
      $j \leftarrow i+1$;
      **while** ~IsCF$(\phi_j, I_{obs})$ **do**
        $j \leftarrow j+1$;
      $\xi_{step} \leftarrow$ CWPnet$(\phi_i, \phi_j, I_{obs})$;
      $\phi_{step} \leftarrow$ Interpolation$(\xi_{step})$; **if** IsCF$(\phi_{step}, I_{obs})$ **then**
        $\phi_{replan} \leftarrow \phi_{replan} \cup \phi_{step}$;
      **else**
        $\phi_{replan} \leftarrow \phi_{replan} \cup$ SWOA$(\left[\phi_i, \phi_j\right], I_{obs})$;
    $i \leftarrow j$;
  **Return** $\phi_{replan}$

Algorithm 1 shows the overall architecture of our proposed online trajectory planning algorithm for HRC. First, the planner obtains the environment encoding information $I_{obs}$ of the workspace in the current state, the robot starting configuration $c_0$ and the target configuration $c_t$. The key obstacle avoidance waypoint $\xi$ of the C-space is calculated by the CWP-net proposed in Section 3.2. Then, the interpolation algorithm

is used to obtain the continuous motion trajectory $\boldsymbol{\phi}$ of each joint. Next, this motion trajectory is determined whether it satisfies the collision-free requirement. If it satisfies, the robot performs the task according to this trajectory. And if it does not satisfy, the initial trajectory is discretized into sampling path points according to the path length.

By iteration, it is judged whether each pair of adjacent sampled path points belongs to the collision-free region or not. The path points in the collision-free interval are retained as part of the replanned trajectory. For the interval where there is collision, first judge whether the target configuration of this interval is located in the collision-free region, and if it is not satisfied, search for the subsequent adjacent collision-free configuration sampling points as the target configuration of the interval.

CWP-net is used again for the replanned interval. However, in some cases, if the neural network cannot find a path that satisfies the obstacle avoidance requirement, the Stochastic waypoint optimization algorithm (SWOA) proposed in Section 4.2 is used for the local optimization of the path in this interval. The above process is repeated in each execution step until the robot reaches the target configuration specified by the task.

To elaborate more specifically, the online trajectory planning algorithm proposed in this paper is implemented by the following methods.

(1) ***CWPnet***: *CWPnet(•)* inputs the starting configuration label, the target configuration label and the obstacle encoding information, and outputs the conformation labels of the discrete obstacle avoidance path points. The output labels need to be decoded to the configurations in C-space. If there are adjacent discrete waypoints with the same label, only the first of them is kept. In order to achieve continuous motion of the robot, the motion intervals between each discrete waypoint also needs to be specified as a series of intermediate waypoints and made to satisfy the motion constraints.

(2) ***Interpolation***: *Interpolation(•)* inputs discrete waypoints and uses a technique to outputs continuous and smooth motion trajectories for each joint based on Quintic B-splines. Quintic B-splines of degree $k$ and order $h = k + 1$ are linear combinations of polynomials $B_{i,k}(u)$ and weighting coefficients $Q_i$. $B_{i,k}(u)$ are generally referred to as the basis functions and $Q_i$ are referred to as the control points. $u$ $(0 \le u \le 1)$ is the B-spline normalized trajectory parameter, where it represents the time variable of the trajectory equation. Thus, the basic form of the B-spline $P(u)$ is as follows.

$$P(u) = \sum_{i=1}^{n} Q_i B_{i,k}(u) \tag{6}$$

where $n$ denotes the number of control points. $B_{i,k}(u)$ satisfies the de Boor–Cox formula [30].

$$\begin{aligned} B_{i,k}(u) &= \frac{u - u_i}{u_{i+k} - u_i} B_{i,k-1}(u) + \frac{u_{i+k+1} - u}{u_{i+k+1} - u_{i+1}} B_{i+1,k-1}(u) \\ B_{i,0}(u) &= \begin{cases} 1 & u_i \le u \le u_{i+1} \\ 0 & elsewhere \end{cases} \end{aligned} \tag{7}$$

where $u_i\,(i = 1, 2, \ldots, n + k + 1)$ is the sequence of nodes. According to the number $v$ of via-waypoints, $v + 12$ nodes are needed to achieve interpolation.

The input sequence of angle-time discrete waypoints is $c_0^j, c_1^j, \ldots, c_{v+1}^j$ $\left(j = 1, 2, \ldots, n_j\right)$. where $j$ denotes the $jth$ joint, determined by the number of joints to be smoothed $n_j$. The trajectory interpolation problem for the $jth$ joint is defined as a planning problem satisfying the following constraints.

$$\begin{cases} P\left(u_0\right)|_{t=0} = \theta_0 \\ \dot{P}\left(u_0\right)|_{t=0} = 0 \\ \ddot{P}\left(u_0\right)|_{t=0} = 0 \\ P\left(u_{v+1}\right)|_{t=f} = \theta_0 \\ \dot{P}\left(u_{v+1}\right)|_{t=f} = 0 \\ \ddot{P}\left(u_{v+1}\right)|_{t=f} = 0 \\ P\left(u_k\right) = \theta_k, (k = 1, 2, \ldots, v) \end{cases} \tag{8}$$

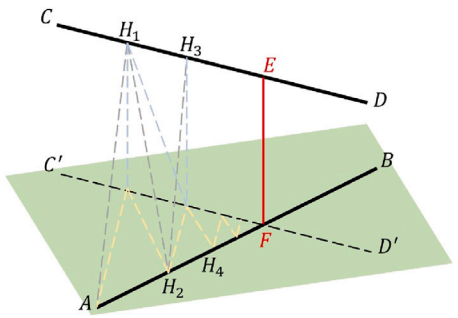



**Fig. 5.** Minimum distance calculation for collision detection.

where $\dot{P}(u_0)|_{t=0}$ and $\ddot{P}(u_0)|_{t=0}$ are the velocity and acceleration constraints at the starting waypoint. $\dot{P}(u_{v+1})|_{t=f}$ and $\ddot{P}(u_{v+1})|_{t=f}$ are the velocity and acceleration constraints at the ending waypoint. $P(u_k) = \theta_k$, $(k = 0, 1, \ldots, v + 1)$ constrains the curve through each waypoint. For the planning problem with action execution time $f$, a mature and efficient solver [31] is used. The collocation matrix is calculated adaptively based on the number of waypoints and constraints, and then the designed linear system is solved using QR factorization.

(3) ***IsCF***: is an abbreviation for *IsCollisionFree(•)*. This function inputs the continuous trajectory planned in the previous steps and outputs the result whether the trajectory meets a collision or not. The input trajectory is first discrete sampled into small steps, and then a line segment-based distance measurement algorithm is used to verify that each discrete configuration is in collision-free space. As shown in Fig. 5, the robot links and the human arm are abstracted into two spatial line segments $\overrightarrow{AB}$ and $\overrightarrow{CD}$, and the function goal is the minimum distance $|EF|$ between them. By adding auxiliary lines, the vertical line of $\overrightarrow{CD}$ is drawn from point $A$, and the vertical foot is the point $H_1$. Similarly, the vertical line of $\overrightarrow{AB}$ is made with $H_1$ as the initial point and the foot point is $H_2$. Repeat the process and the minimum distance $|EF| = \left|H_n H_{n+1}\right| < \cdots < \left|H_2 H_3\right| < \left|H_1 H_2\right|, (n \subset N^*)$ will be obtained by iteration. The spatial relations and theoretical proofs of the line segments are given in the literature [32]. With this method, it is possible to efficiently obtain the distance between the robot's three moving links and the human arm, and subsequently control the radius and safety threshold of the cylindrical envelopes of both.

### 4.2. An improved STOMP for optimization

The STOMP algorithm [8] is a method for motion planning using random trajectories, which explores the space around the initial trajectory by generating noise path points to find a lower-cost trajectory. This method does not require gradient information, and its stochastic nature is beneficial to overcome local minima. However, STOMP also has some limitations, such as the need for an ideal initial trajectory and the lack of direction of the stochastic exploration process. For each path point, the covariance matrix of the generated noise distribution is the same, leading to a consistent exploration range. This existence of redundant exploration areas introduces bias and affects the planning speed and results.

Considering the above limitations of the original algorithm, an improved STOMP algorithm, called Stochastic waypoint optimization algorithm (*SWOA(•)* in Algorithm 1), is proposed in this paper for optimizing the undesirable parts of the initial trajectory. The specific implementation process of this algorithm is as follows.

The proposed improved STOMP algorithm is used to optimize the input joint trajectory, which has a starting configuration of $\boldsymbol{\phi}_i$ and an ending configuration of $\boldsymbol{\phi}_j$. The trajectory is sampled in equal steps

**Algorithm 2:** Stochastic waypoint optimization algorithm

**input :** ∘ $[\boldsymbol{\phi}_i, \boldsymbol{\phi}_j]$ – Initial joint trajectory that starts with $\boldsymbol{\phi}_i$ and ends with $\boldsymbol{\phi}_j$;
∘ $\boldsymbol{I}_{obs}$ – Obstacle Information;
**output:** ∘ $\boldsymbol{\phi}_{replan}$ – Collision-free trajectory;
**given :** ∘ $\boldsymbol{\Omega}$ – A positive semi-definite matrix representing control costs;
∘ $\boldsymbol{S}$ – A smooth matrix;
∘ $\boldsymbol{\mu}_{k_m}, \boldsymbol{\Sigma}_{k_m}$ – From $DPGMM(\boldsymbol{\psi}_m)$, $k_m = 1, 2, \cdots, N_{k_m}, m = 1, 2, 3$.

1 $\{\boldsymbol{\phi}_1, \boldsymbol{\phi}_2, \cdots, \boldsymbol{\phi}_{N_s}\} \leftarrow$ `Sample(`$\boldsymbol{\phi}_i, \boldsymbol{\phi}_j$`)`;
2 $\boldsymbol{\theta}_\gamma = \{\theta_{\gamma,1}, \theta_{\gamma,2}, \cdots, \theta_{\gamma,N_s}\}$ ←*Extract the angle sequence of joint* $\gamma$ ;
3 $\boldsymbol{k}_m = \{\kappa_1, \kappa_2, \cdots, \kappa_{N_s}\}$ ←*Search for the sub-model tag vector of* $\boldsymbol{\theta}_\gamma$ ;
4 **Repeat until convergence of** $C_{total}(\boldsymbol{\theta})$
5  **for** $\alpha \leftarrow 0$ **to** $N_s$ **do**
6  $\{\boldsymbol{\varepsilon}^1_{\gamma,\alpha}, \boldsymbol{\varepsilon}^2_{\gamma,\alpha}, \cdots, \boldsymbol{\varepsilon}^M_{\gamma,\alpha}\} \leftarrow \mathcal{N}(\boldsymbol{\mu}_{\kappa_\alpha}, \boldsymbol{\Sigma}_{\kappa_\alpha})$;
7  **for** $\beta \leftarrow 0$ **to** $M$ **do**
8  $P(\boldsymbol{\varepsilon}^\beta) \leftarrow \frac{exp[-C(\boldsymbol{\varepsilon}^\beta)/\lambda]}{\sum_{p=1}^{M} exp[-C(\boldsymbol{\varepsilon}^p)/\lambda]}$;
9  **for** $\alpha \leftarrow 1$ **to** $N_s - 1$ **do**
10  $\delta\boldsymbol{\varepsilon}_\alpha \leftarrow \sum_{p=1}^{M} P(\boldsymbol{\varepsilon}^p_\alpha)\delta(\boldsymbol{\varepsilon}^p_\alpha - \boldsymbol{\mu}^\omega_{\kappa_\alpha})$;
11  $\boldsymbol{\theta} \leftarrow \boldsymbol{\mu}^\omega_{\kappa_\alpha} + \boldsymbol{S}\delta\boldsymbol{\varepsilon}$;
12  $C_{total}(\boldsymbol{\theta}) \leftarrow \sum_{\alpha=1}^{N_s} c(\theta_\alpha) + \frac{1}{2}\boldsymbol{\theta}^T\boldsymbol{\Omega}\boldsymbol{\theta}$;
13  **for** $\alpha \leftarrow 0$ **to** $N_s$ **do**
14  $[\boldsymbol{R}, \boldsymbol{\Gamma}] \leftarrow$ `EVD(`$\boldsymbol{\Sigma}_{\kappa_\alpha}$`)`;
15  $(a_\alpha, b_\alpha) \leftarrow 2\sqrt{2}(\sqrt{\boldsymbol{\Gamma}_{2,2}}, \sqrt{\boldsymbol{\Gamma}_{1,1}})$;
16  $(x_0, y_0) \leftarrow \boldsymbol{R}^T\boldsymbol{\theta}_\alpha - \boldsymbol{R}^T\boldsymbol{\mu}_{\kappa_\alpha}$;
17  $(a^*_\alpha, b^*_\alpha) \leftarrow$ `Update(`$x_0, y_0, a_\alpha, b_\alpha$`)`;
18  $\boldsymbol{\mu}_{\kappa_\alpha} \leftarrow \boldsymbol{\theta}_\alpha$;
19  $\boldsymbol{\Sigma}_{\kappa_\alpha} \leftarrow \boldsymbol{R}\left[b^{*2}_\alpha/8, 0; 0, a^{*2}_\alpha/8\right]\boldsymbol{R}^T$;
20 ***Return*** $\boldsymbol{\phi}_{replan}$;

using *Sample(•)* to obtain a sequence of discrete waypoint configurations $\{\phi_1, \phi_2, \ldots, \phi_{N_s}\}$. At this point, the sequence of path points corresponding to the joint $\gamma$ is $\boldsymbol{\theta}_\gamma = \{\theta_{\gamma,1}, \theta_{\gamma,2}, \ldots, \theta_{\gamma,N_s}\}$. According to the DPGMM distribution model in Section 3.1, the sequence of submodel labels $\boldsymbol{k}_m = \{\kappa_1, \kappa_2, \ldots, \kappa_{N_s}\}$ to which the sequence of path points belongs is obtained. The labels of joints 1 and 2 are based on $\Lambda^1_{k_1}, (k_1 = 1, 2, \ldots, N_{k_1})$, the labels of joints 3 and 4 are based on $\Lambda^2_{k_2}, (k_2 = 1, 2, \ldots, N_{k_2})$, and the labels of joints 5 and 6 are based on $\Lambda^3_{k_3}, (k_3 = 1, 2, \ldots, N_{k_3})$. In the iteration of Algorithm 2, $M$ noise trajectories $\{\boldsymbol{\varepsilon}^1_{\gamma,\alpha}, \boldsymbol{\varepsilon}^2_{\gamma,\alpha}, \ldots, \boldsymbol{\varepsilon}^M_{\gamma,\alpha}\}$ are generated from $\mathcal{N}(\boldsymbol{\mu}_{\kappa_\alpha}, \boldsymbol{\Sigma}_{\kappa_\alpha})$ depending on the labels of each path point.

In the 8th line of Algorithm 2, the probability $P(\boldsymbol{\varepsilon}^\beta)$ [8] is given for calculating each noise trajectory for each waypoint, where the exponential term is calculated as

$$exp\left[-C\left(\boldsymbol{\varepsilon}^\beta\right)/\lambda\right] = exp\left\{\frac{-\rho\left[C\left(\boldsymbol{\varepsilon}^\beta\right) - \min C\left(\boldsymbol{\varepsilon}^\beta\right)\right]}{\max C\left(\boldsymbol{\varepsilon}^\beta\right) - \min C\left(\boldsymbol{\varepsilon}^\beta\right)}\right\} \tag{9}$$

where $C(\boldsymbol{\varepsilon}^\beta)$ is the path cost function, $\lambda$ is the adaptively adjusted cost sensitivity parameter, and $\rho$ is an artificially set constant.

In the 10th line, the probability weighted sum of noise parameters is calculated to update each joint motion of discrete waypoints in the 11th line. The updated path cost $C_{total}(\boldsymbol{\theta})$ is calculated from line 12.

$$C\left(\boldsymbol{\theta}_\alpha\right) = \sum_{t=0}^{\alpha} C_{obs} + C_{len} + C_{con} \tag{10}$$

where $C_{obs} = \sum_{l\in\mathcal{L}}\left[\left\|\dot{x}_l\right\| \max\left(\varepsilon + r_l - d\left(x_l\right), 0\right)\right]$ denotes the obstacle cost. The robot link is $l \in \mathcal{L}$ and $d(\bullet)$ is the signed Euclidean distance transform (EDT) [8] based on the discrete workspace of Section 2. $\varepsilon$ is the minimum safe distance, $r_l$ is the radius of the link cylindrical envelope. $x_l$ is the link's position and $\dot{x}_l$ is its velocity of motion. $C_{len}$ denotes the path length cost, which is approximated as the sum of the Euclidean distances between each adjacent path point of the end-effector. $C_{con}$ denotes the constraint cost, which can be satisfied by adding a degree to the cost function, describing the distance to the constraint boundary for joint angle restrictions, speed restrictions and end-effector orientation constraints, etc.

It is worth noting that in this paper the noise path points generated by $\mathcal{N}(\boldsymbol{\mu}_{\kappa_\alpha}, \boldsymbol{\Sigma}_{\kappa_\alpha})$ are constrained to the path distribution model in Section 3.1, which has parameters learned from sample data that satisfy the joint constraints. Therefore, the path exploration within the noise feasible range does not exceed the joint constraints. The velocity can also be constrained by controlling the time step and the discrete path trajectory step. Furthermore, $\boldsymbol{\theta}^T\boldsymbol{\Omega}\boldsymbol{\theta}$ in line 12 denotes the path acceleration squared term, where $\Omega$ is a positive semi-definite matrix representing control costs, computed from $\boldsymbol{\Omega} = \boldsymbol{A}^T\boldsymbol{A}$. The smoothing matrix $\boldsymbol{S}$ takes the value $\boldsymbol{R}^{-1}$. $\boldsymbol{A}$ is a finite difference matrix [8] as follow.

$$\boldsymbol{A} = \begin{bmatrix} 1 & 0 & 0 & & 0 & 0 & 0 \\ -2 & 1 & 0 & \cdots & 0 & 0 & 0 \\ 1 & -2 & 1 & & 0 & 0 & 0 \\ & \vdots & & \ddots & & \vdots & \\ 0 & 0 & 0 & & 1 & -2 & 1 \\ 0 & 0 & 0 & \cdots & 0 & 1 & -2 \\ 0 & 0 & 0 & & 0 & 0 & 1 \end{bmatrix}$$

Lines 13 to 19 are the optimal exploration region update process. For each path point, the eigenvalue decomposition $EVD(\boldsymbol{\Sigma}_{\kappa_\alpha})$ is performed for the distribution sub-model parameter $\boldsymbol{\Sigma}_{\kappa_\alpha}$ to which it belongs. The decomposition yields $\boldsymbol{\Sigma}_{\kappa_\alpha} = \boldsymbol{R}\boldsymbol{\Gamma}\boldsymbol{R}^T$, where $\boldsymbol{\Gamma}$ is the eigenvalue matrix from which the noise path exploration range can be calculated by line 15. The boundary of this range takes the form of an ellipse with semi-long axis $a_\alpha$ and semi-short axis $b_\alpha$. $\boldsymbol{R}$ is the unit orthogonal matrix, which represents the rotation matrix of the boundary ellipse with respect to the basic coordinate system. The coordinates of the boundary ellipse center in the original exploration range coordinate system are obtained from line 16.

In order to optimize the exploration range after each iteration step, this paper proposes an iterative method $Update(\bullet)$ for quickly solving the elliptic boundary parameters $a^*_\alpha, b^*_\alpha$ required for the next iteration in line 17. As shown in Fig. 6, the method is able to solve the problem expressed as the known parameters of the boundary ellipse $E_1$ (center coordinate $(x^0_1, y^0_1)$, semi-long axis length $a_1$, and semi-short axis length $b_1$) and the center coordinate $(x^0_2, y^0_2)$ of the updated inscribed ellipse $E_2$, solving for the semi-long axis length $a_2$ and semi-short axis length $b_2$ of the boundary ellipse $E_2$. The updated boundary ellipse is specified as a similar ellipse to the original boundary ellipse, so that the interior tangent point $P_t(x_t, y_t)$ satisfies the condition: $\|x_t - x^0_2\|/a_2 = \|x_t - x^0_1\|/a_1$.

Taking Fig. 6(b) as an example, the position of the tangent point is found in the interval $[-sqrt(a^2_2[1 - (y^0_2)^2/b^2_2]), x^0_2]$. First, the interval is discretized in equal steps as $[x_0, x_1, \cdots, x_N]$, and the $E_2$ parameter is solved in turn according to the inner tangent scaling relation when the coordinates of the tangent point are assumed to be $x_p = x_1, \ldots, x_{N-1}$. Then, the $E_2$ coordinates $P_l(x_{p-1}, y_{p-1})$ and $P_r(x_{p+1}, y_{p+1})$ corresponding to the neighboring points $x_{p-1}$ and $x_{p+1}$ of the point $x_p$ are obtained. Substituting the two points $P_l$ and $P_r$ into $E_1$, $\epsilon_{p-1} = x^2_{p-1}/a^2_1 + y^2_{p-1}/b^2_1$ and $\epsilon_{p+1} = x^2_{p+1}/a^2_1 + y^2_{p+1}/b^2_1$ are calculated. From Fig. 6(c), $\epsilon_{p-1}$ and $\epsilon_{p+1}$ are both smaller than 1 when $x_p$ is a tangent point. In Fig. 6(d), when $x_p$ is not a tangent point, one of $\epsilon_{p-1}$ and $\epsilon_{p+1}$ is larger than 1 and the other is smaller than 1. Note that our method is an approximate method, and the parameters of the inner tangent ellipse obtained are related to the number of samples $N$ in the interval. There exists the case shown in Fig. 6(e), where there exist two intersections of ellipse $E_1$ and ellipse

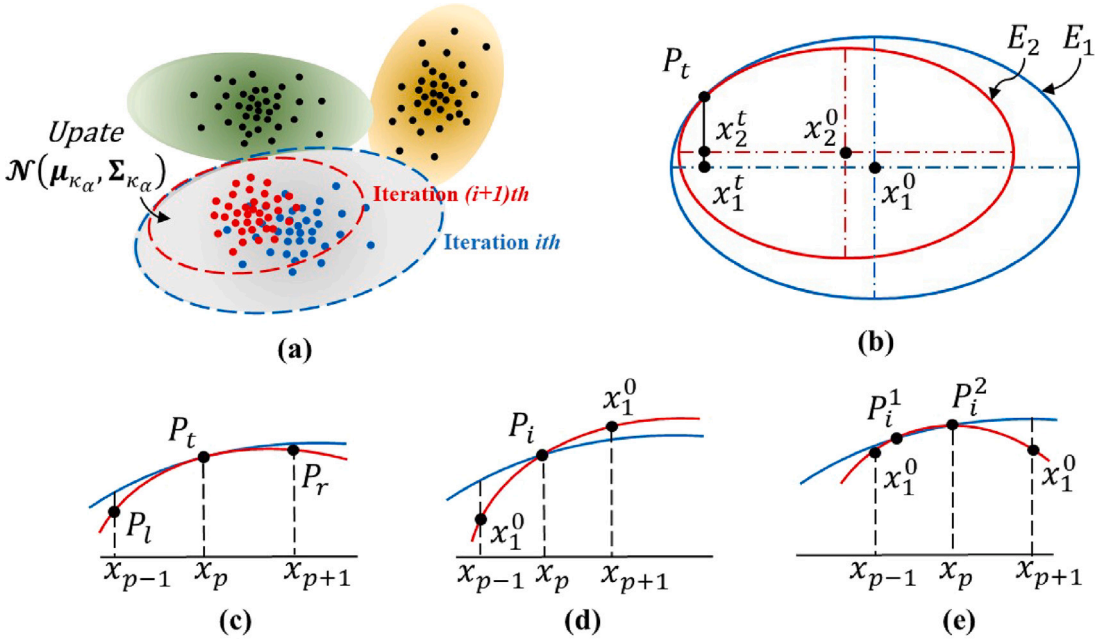



**Fig. 6.** Noise exploration boundary ellipse update rule schematic. (a) The DPGMM distribution sub-model constraint range. (b) Simplified model of the boundary ellipse update rule. (c) The state of tangency. (d) The state of intersection. (e) Special state due to discrete accuracy.

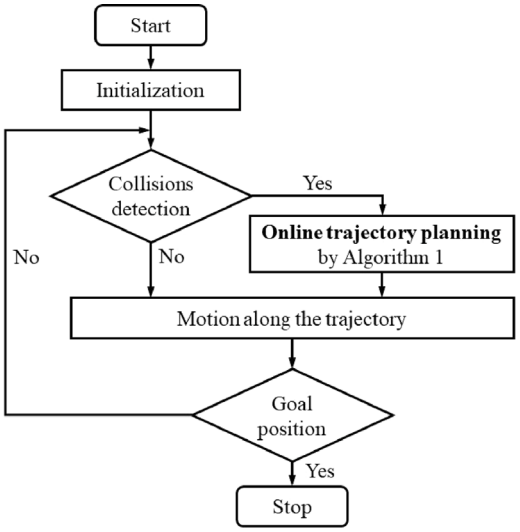



**Fig. 7.** Flow chart of the proposed safety trajectory generation algorithm.

$E_2$ in discrete steps, satisfying the condition that both $\epsilon_{p-1}$ and $\epsilon_{p+1}$ are less than 1 at the same time. Therefore, the derived approximate inner tangent ellipse parameters are a combination of accuracy and computational efficiency. In this paper, the number of discretization is selected as 100, the computation time is less than 1 ms, and the computation efficiency is higher than that of the analytical solution method for solving the system of parametric equations, which can meet the demand of real-time computation.

Based on the above solution process, $\boldsymbol{\mu}_{\kappa_\alpha}$ and $\boldsymbol{\Sigma}_{\kappa_\alpha}$ for the next iteration step is updated by lines 17 and 18. Algorithm 2 iterates until the total cost of the path $C_{total}(\theta)$ converges. To summarize, the proposed dynamic trajectory planning process is shown in Fig. 7.

## 5. Experiments and result analysis

In this section, the effectiveness of the proposed online collision-free trajectory generation algorithm is demonstrated by simulations and real experiments. The dynamic process of HRC is simulated based on a real industrial case. The results of the experiments verify that the algorithm has the ability to actively avoid collision with the human in the shared workspace in both static and dynamic situations.

### 5.1. An industrial use case

We validate the approach proposed in this paper with an industrial use case of HRC applied in the field of aircraft final assembly testing. The working environment is an aircraft cockpit. During the collaborative process, the human operator seats in the captain pilot seat of the aircraft cockpit and the collaborative robot is located in the co-pilot area. The collaborative task is to operate the switches, buttons, and manipulation mechanisms (including the driver's wheel, driver's column, and footrest) in the relevant area according to the test procedure, and cooperate to complete the test task. The schematic diagram of our proposed test system is shown in Fig. 8.

In this use case, the discrete resolution $D$ is determined by Eq. (1) based on the shared workspace dimensions and taken as 5. Thus, the shared workspace is discretized into $5 \times 5 \times 5$ voxels so that it completely covers the space of possible interference with the operator during motion. The robot can perform collaborative operations in this space in known areas such as the central control panel, the display control screen and the top control panel. The typical HRC configurations in this case are shown in Fig. 9.

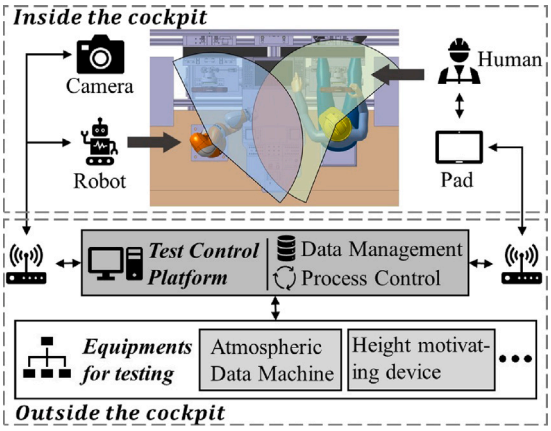



**Fig. 8.** An industrial use case of collaborative robot-assisted execution solution for airborne system testing.

**Table 2**
Initial task setup and collision-free intermediate configurations generated by the proposed algorithm.

| | Joint | | | | | |
|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 |
| Static experiment 1 | | | | | | |
| $c_0$ | 1.2113 | 0.8761 | −0.0670 | −1.6991 | 0.6956 | −0.8617 |
| $c_1$ | 1.3499 | 0.8159 | 0.1203 | −1.5880 | 0.7562 | −0.8682 |
| $c_2$ | 1.4245 | 0.3546 | 0.1203 | −1.5880 | 0.7562 | −0.8682 |
| $c_3$ | 1.4245 | 0.3546 | 0.6775 | −1.6534 | 0.7562 | −0.8682 |
| $c_t$ | 1.8884 | 0.4479 | 0.2962 | −2.0297 | 0.7582 | −0.8601 |
| Static experiment 2 | | | | | | |
| $c_0$ | 2.3972 | 0.9928 | 0.2963 | −1.3596 | 1.2952 | −0.0835 |
| $c_1$ | 1.6895 | 0.9701 | 0.2847 | −1.2307 | 1.0320 | −0.0503 |
| $c_t$ | 1.3503 | 0.8371 | 0.2468 | −1.9875 | 1.2945 | −1.6858 |
| Static experiment 3 | | | | | | |
| $c_0$ | 1.2987 | 0.8547 | 0.2413 | −1.6388 | 0.6793 | −1.1404 |
| $c_1$ | 1.3499 | 0.8159 | 0.2167 | −1.2660 | 0.8346 | −1.5343 |
| $c_2$ | 1.3499 | 0.8159 | 1.3641 | −1.8566 | 0.8346 | −1.5343 |
| $c_t$ | 1.4920 | 1.1075 | 0.9255 | −1.6115 | 0.3877 | −1.3558 |

### 5.2. Collaborative simulation

In this section, simulations are designed to evaluate the proposed algorithm and compare its performance with the state-of-the-art classical planners. The HRC task considered in the study is a robot-assisted test system in the cockpit of an aircraft.

#### 5.2.1. Implementation details

The test control platform for simulation is a desktop computer configured with Intel Core i7-8700 processor at 3.20 GHz. Our system uses a KUKA LBR iiwa 14 robot to perform in-cabin manipulation tasks with a human operator. For different task and obstacle environment configurations, the robot motion trajectories generated by the classical planner and the motion trajectories from historical tasks are used as empirical trajectories in the simulations, and a total of 1209 sample data are finally identified. Empirical distribution learning is performed on these sample data, and the DPGMM models of the three robot links are obtained according to the method described in Section 3.1. A total of 22 submodels $\Lambda^1_{k_1}, (k_1 = 1, 2, \dots, 22)$ are generated for the distribution model of Link 1, 19 submodels $\Lambda^2_{k_2}, (k_2 = 1, 2, \dots, 19)$ are generated for the distribution model of Link 2, and 21 submodels $\Lambda^3_{k_3}, (k_3 = 1, 2, \dots, 21)$ are generated for the distribution model of Link 3. The distribution of DPGMM sub-models for each link is shown in Fig. 10. Based on this distribution model, the normalized configuration labels of the sample path points are constructed by the methods in Section 3.

The designed CWP-net is implemented with keras [33], which uses a fully connected network structure with four hidden layers, where the number of neurons in each layer is (127, 85, 57, 37, 23, 10), with the ReLU activation functions and dropout layers with dropout rate p = 0.3 except the last layer. In the training process, we use the Adam optimizer with learning rate $\lambda = 0.001$ to optimize the weight parameters, and set the batch size to 64. In addition, we use the data augmentation technique to enlarge our training set for better generalization and robustness. That is, we randomly adjust the obstacle pose of the original sample path to generate new obstacles that match the collision-free path near by original voxels. Finally, total of 6045 training trajectories are obtained. In Appendix B, we elaborate the source of the training data and the data augmentation techniques. The network hyperparameter selection process, the uniformity of the training data distribution and the generalization of the model are all verified by K-fold cross-validation experiments in Appendix C. The loss of the best test model converges after 112 epoches at loss = 0.02538 in 47.99s on a Nvidia Tesla V100S GPU.

#### 5.2.2. Static simulation without human motion

In this part, we simulate three static configurations common to the collaborative process. Using the proposed algorithm, trajectories are automatically generated in which the robot can accomplish the task goal while avoiding the arms of its human colleagues. Besides, the position, velocity, and acceleration profiles in the joint space of each trajectory are demonstrated. Since our goal is to plan obstacle avoidance trajectories and the robot's seventh axis angle does not affect the robot's obstacle avoidance pose, the evaluated obstacle avoidance configuration consists of angles from 1 to 6 joints.

Fig. 11 shows the task configuration and simulation results for static experiment 1. The starting configuration $c_0$ and the ending configuration $c_t$ for this task are shown in the static experiment 1 of Table 2. A randomly generated envelope cylinder simulating a human arm operating the central control panel is used to represent the obstacle. To avoid collisions during robot motion, the proposed algorithm automatically generates three intermediate configurations $c_1$, $c_2$ and $c_3$, listed in Table 2. To show the obstacle avoidance process more clearly, the start, end and intermediate path point configurations of the planned trajectory are given in Fig. 11(a), and the end-effector motion trajectory is shown in a green line. The discrete configuration of the entire trajectory is shown in Fig. 11(b), which clearly shows that the proposed method can guarantee the distance between the robot body and the obstacles during the whole motion. Similarly, Fig. 12 shows the setting and results of static experiment 2. To avoid collisions, the algorithm automatically generates an intermediate configuration $c_1$. The specific parameters are shown in the static experiment 2 part of Table 2. The result of another statics experiment 3 is shown in Fig. 13. In this obstacle and task condition, the algorithm generates two intermediate configurations. The configuration data are shown in the experiment 3 part of Table 2.

From the above three static simulations, it can be seen that the proposed algorithm can effectively and automatically generate the path points required for obstacle avoidance for the environmental configuration of HRC in the cockpit, and plan the trajectory of the executable task without interfering with the human task. The trajectory generated by the algorithm not only ensures that the robot end-effector does not cause a collision, but also takes into account that each link of the robot body does not collide with the obstacles. At the same time, the method in this paper achieves the goal of trajectory planning in the joint space. The generated trajectories are continuous and smooth in both velocity and acceleration levels. The starting point, ending point and motion process of the trajectory satisfy the velocity and acceleration constraints specified by the task.

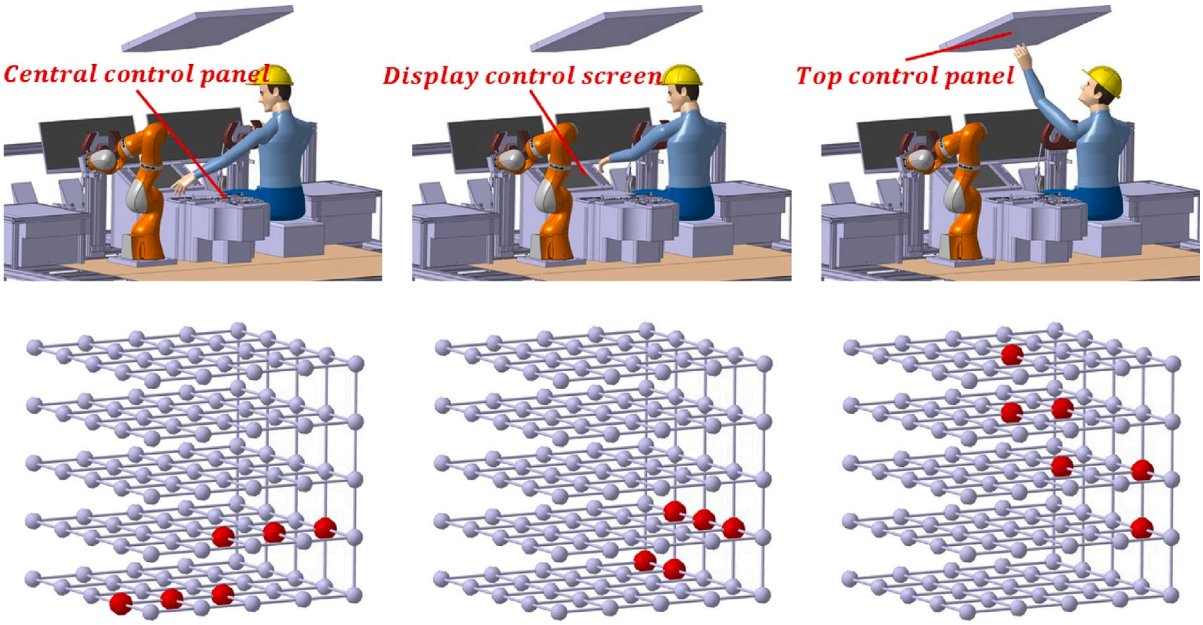



**Fig. 9.** Examples of human forms in the discrete shared workspace. (The red dots represent the subspaces occupied by the human arm as an obstacle. The gray dots represent the free subspaces in which the robot can plan movements). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

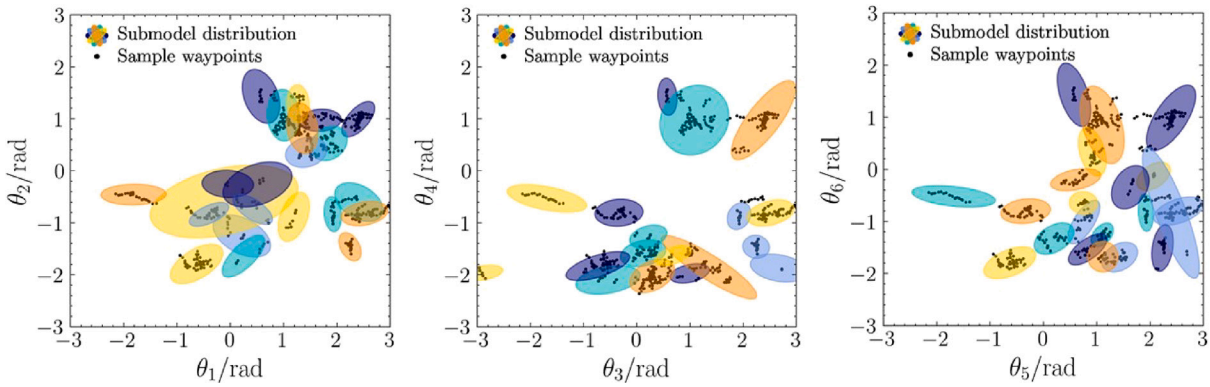



**Fig. 10.** Waypoint distribution models of robot sample obstacle avoidance trajectories using DPGMM. (a) Distribution sub-model of Link 1. (b) Distribution sub-model of Link 2. (c) Distribution sub-model of Link 3.

It is worth noting that the proposed CWP-net planned collision-free path points are trained based on the DPGMM distribution sub-model. To a certain extent, the planned trajectories lose a part of the accuracy of the sample trajectories, and there exists a risk of collision with obstacles in the trajectories generated by the network model. Therefore, we propose an improved STOMP-based SWOA optimization method in Section 4.2 for replanning when the neural network planning fails or the trajectory is unsatisfactory. Fig. 14(a) shows the obstacle avoidance trajectory generated directly by CWP-net in one experiment, and it can be seen that the three generated intermediate path points have one path point with large deviation. Thus, the SWOA algorithm is used to optimize this part in the path segment where the collision is occurred. The replanned outcome is shown in Fig. 14(b), and it can be seen that the replanned trajectory can fully satisfy the obstacle avoidance demand. Overall, the combination of CWPnet+SWOA is a compromise consideration of computational efficiency and trajectory quality, which can effectively make the online trajectory planner balance planning efficiency and collision avoidance robustness.

#### 5.2.3. Algorithm efficiency comparison

In this part, we present the results of the planning time evaluation of the proposed algorithm with the state-of-the-art classical planners: the RRT-connect, RRT* and STOMP. We use the standard Open Motion Planning Library (OMPL) [34] to implement the classical planners. The experimental environment is built by MoveIt [35], as shown in Fig. 15. The robot workspace is surrounded by fixed obstacles such as simulated cockpit wall board, overhead control panel, central control panel, display control screen, and driving pillar. Cylinders are used to simulate the envelope of the human arm and different postures are constructed randomly. 30 simulated task scenarios different from the training data are created, and the task objectives of this experiment refer to the actual operational requirements. In Fig. 16, we report the test data for these 30 experiments. In all planning problems:

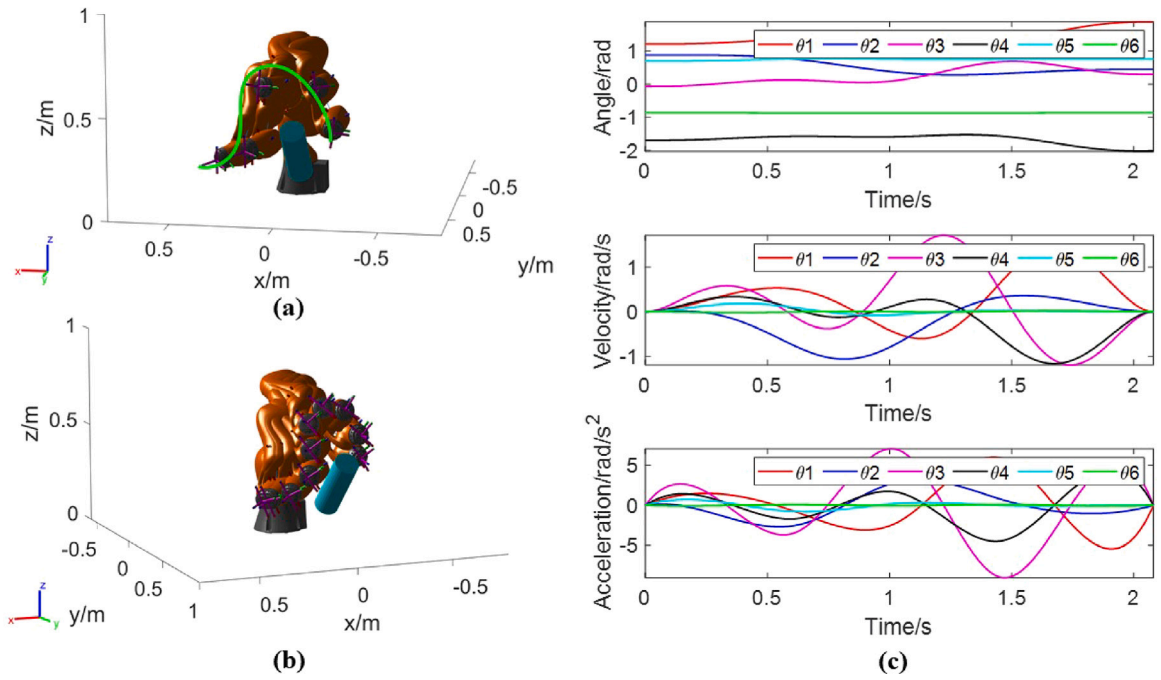



**Fig. 11.** Static obstacle avoidance simulation configuration 1. (a) The critical waypoint configuration and end-effector motion trajectory generated by the proposed algorithm. (b) Schematic of the discrete sampling trajectory during the motion. (c) Joint angle, velocity and acceleration of the obstacle avoidance trajectory.

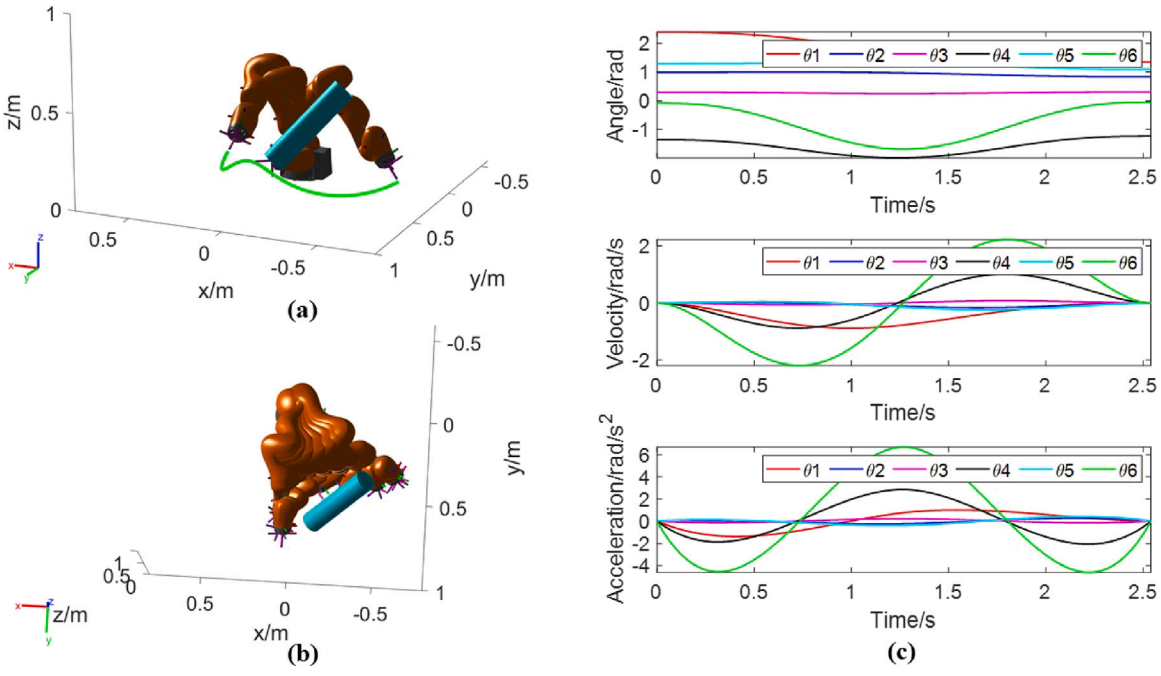



**Fig. 12.** Static obstacle avoidance simulation configuration 2. (a) The critical waypoint configuration and end-effector motion trajectory generated by the proposed algorithm. (b) Schematic of the discrete sampling trajectory during the motion. (c) Joint angle, velocity and acceleration of the obstacle avoidance trajectory.

- The average computation time of RRT-connect is 1.1176 s with a standard deviation of 0.1244 s.
- The average computation time of RRT* is 4.3410 s with a standard deviation of 1.3903 s.
- The average computation time of STOMP is 1.4901 s with a standard deviation of 0.4296 s.

And according to the process of our proposed Algorithm 1, the trajectories directly planned by CWP-net are feasible for 16 (53%) in the total experiments. This part of the results is not optimized by the SWOA algorithm, and the average planning time is 0.0094 s with a standard deviation of 0.00028 s. In contrast, there are 14 (47%) paths planned by CWP-net that are optimized by SWOA, and the average

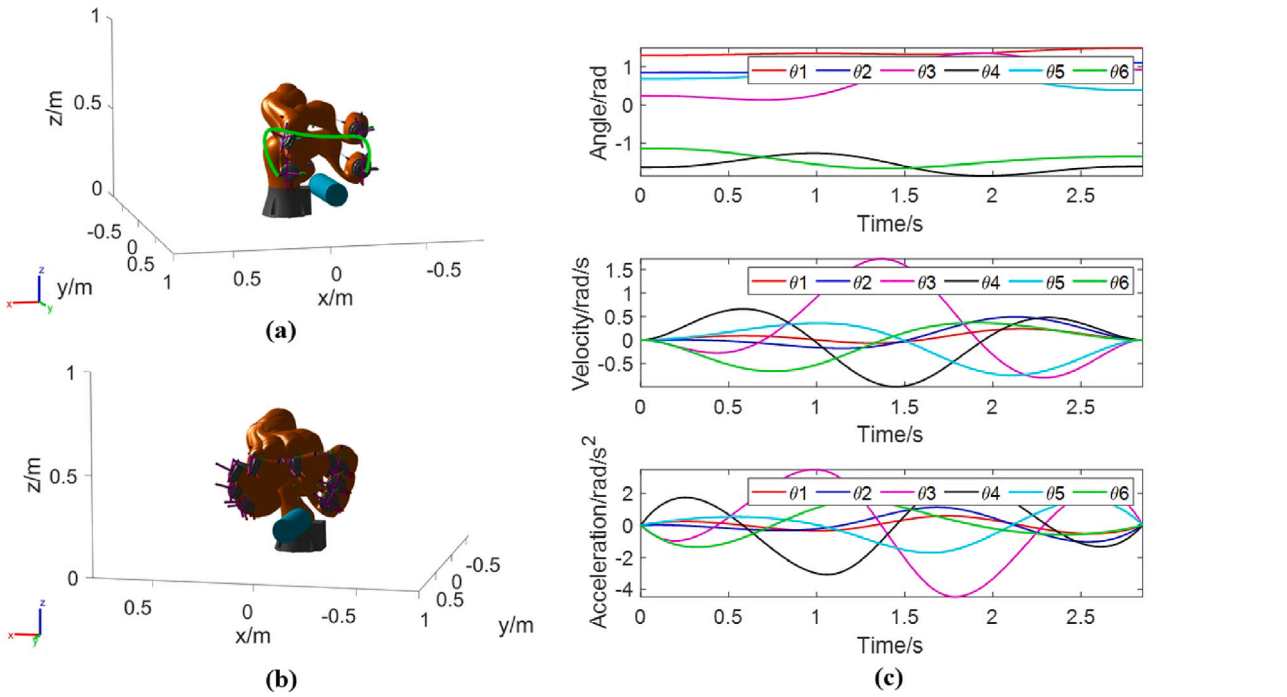



**Fig. 13.** Static obstacle avoidance simulation configuration 3. (a) The critical waypoint configuration and end-effector motion trajectory generated by the proposed algorithm. (b) Schematic of the discrete sampling trajectory during the motion. (c) Joint angle, velocity and acceleration of the obstacle avoidance trajectory.

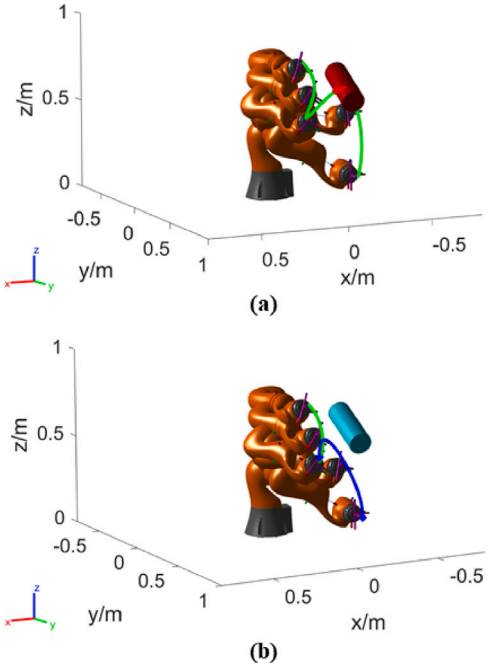



**Fig. 14.** Example of the proposed SWOA trajectory optimization. (a) Trajectory planned by CWP-net. (b) Trajectory output after SWOA replanning.

planning time for this part is 0.5877 s with a standard deviation of 0.1621 s. The overall average time for the two parts is 0.2793 s.

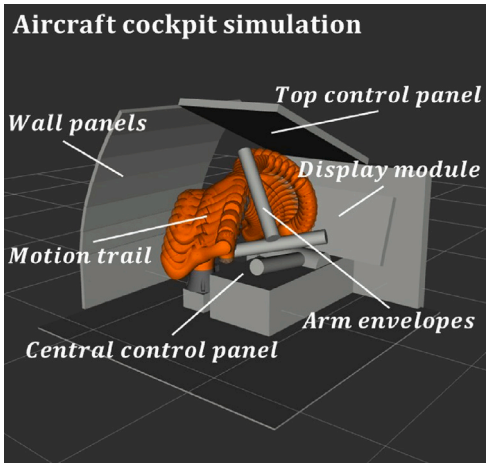



**Fig. 15.** Simulation environment for algorithm comparison experiments.

From the results, it can be analyzed that the proposed algorithm is 4.0 times and 15.5 times faster than RRT-connect and RRT*, respectively. Also the improved STOMP combined with CWP-net algorithm is at least 2.5 times faster than the original STOMP algorithm. Also, in the simulation, the average number of iterations for STOMP to converge to the path requirement is 43.6, while with the improved SWOA algorithm, the average number of experimental iterations is 29.3. The improved algorithm converges significantly faster to the feasible range, proving the effectiveness. Increasing the training samples with full consideration of the environmental configuration can somewhat reduce the number of tasks requiring SWOA optimization and increase the

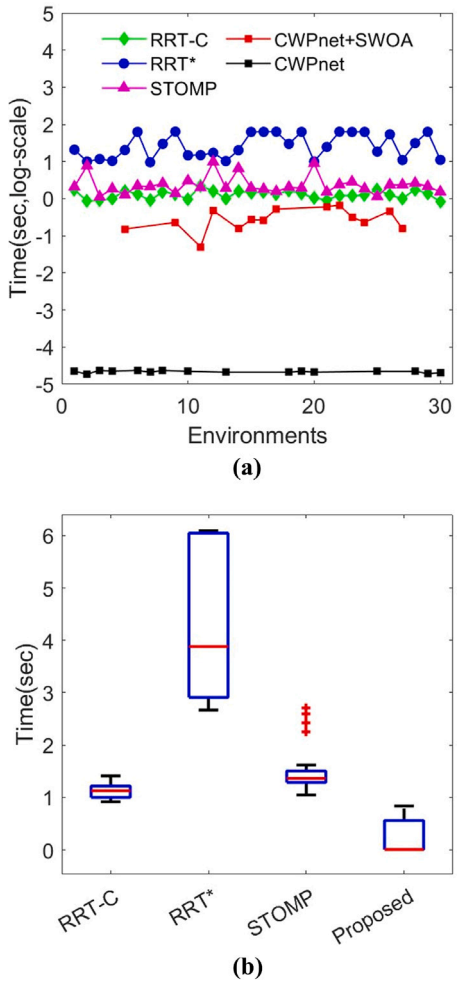



**Fig. 16.** Comparison of experimental results. (a) Planning time of each algorithm for log-scale with different simulation configurations. (b) Distribution of test results.

percentage of CWP-net direct output, which can better improve the average planning efficiency of the algorithm.

The computation time of traditional planners is difficult to meet dynamic planning, and with algorithms such as RRT*, the planning time will increase significantly as the target path cost decreases. For CWP-net, the training process is able to adopt the trajectories planned by any advanced planners with the desired path cost. The computation time of the model stably stays around 10 ms, which is able to meet the demand of dynamic obstacle avoidance. This approach constructs a bridge between an offline planner and an online planning task with some generalizability.

#### 5.2.4. Dynamic simulation with human motion

In this section, we give a simulation experiment under dynamic human arm movements. Fig. 17 provides the simulation results of the dynamic changes of the collaborator's arm during HRC and the online adaptive trajectory of the robot. Fig. 17(a) shows the robot motion trajectory from the initial configuration to the target configuration determined offline in the state of no obstacle interference. When the human arm moves, resulting in Fig. 17(b). The algorithm detects collision interference in the robot trajectory and generates collision-free path point configurations to avoid collisions. The replanned trajectory that can reach the target configuration is shown in the green dashed part in Fig. 17(c). The robot motion trajectory, after real-time correction, moves according to the replanned trajectory as shown in Fig. 17(d). The collaborator moves again and the arm space state changes, as shown in Fig. 17(e). The algorithm replans the smooth and continuous obstacle avoidance trajectory in the joint space online and executes it, as shown in Fig. 17(f)–(h).

### 5.3. Real collaboration scenario testing

The real HRC test environment consists of a collaborative robot, a vision measurement module, and a simulated cockpit. The simulated cockpit consists of a force-sensitive driver's wheel, joystick, footrest and modules such as buttons, switches and displays, which are of equal size to the real environment. The robot is a KUKA iiwa14 collaborative robot, including the robot body, controller PC, smartpad and SCHUNK Co-act EGP-C 40-N-N-KTOE gripper. The vision measurement module consists of two Intel RealSense D435i RGB-D cameras, to obtain multi-angle views in the workspace. An open source project MediaPipe [36] is used to perform arm tracking and positioning. The overall control module of the online trajectory planning algorithm and the vision module are built on a separate server-side program on a PC to control the robot-side controller in real time via TCP/IP protocol.

Fig. 18(a) shows task configuration 1 for the simulated cabin collaboration scenario, and the green line shows the default motion trajectory of the robot in the undetected obstacle condition. The starting state is located near the display control panel and the target state is located at the outer end of the central control panel. The entire end-effector motion path is parallel to the central axis of the cockpit. When the collaborator arm moves to the central control panel region for operation, as in Fig. 18(b), the algorithm generates path points online to replan the trajectory in order to avoid collisions. The robot moves above the arm and then crosses the arm to finally reach the target position according to the replanned path, and the process is shown in Fig. 18(c) and (d). Figure e gives another task configuration 2 for the collaborative scenario. The collaborator arm is located at the height of the speed brake handles to simulate the process of operating the handles, and the default trajectory and replanned trajectory of the robot from the starting position at the display control panel to the end of the central control panel are shown in Fig. 18(e). Fig. 18(f) shows the collaborative scenario task configuration 3. The robot starting state is located at the end of the central control panel, and the target state is the top control panel area. During the robot's movement along the default motion path, the collaborator arm moves to the top control panel area for operation. After the system detects that the arm interferes with the initial path, it generates a collision-free configurations as in Fig. 18(g) and (h). The robot bypasses the collaborator arm to reach the target location without interfering with the work of the collaborator, as shown in Fig. 18(i).

The above process tested a typical configurations of the collaboration process. The system's autonomously replanned obstacle avoidance trajectories are shown in Fig. 19. The test results demonstrate the ability of the proposed algorithm to avoid human–robot collisions in collaborative aircraft assembly testing or similar collaborative shared workspace production activities.

## 6. Conclusions

In this paper, a new collaborative human–robot safety trajectory generation method is proposed for industrial applications with shared workspaces. The proposed robot collision avoidance trajectory planning algorithm is applicable to dynamic environments. The algorithm consists of two parts. In the first part, we first design the Collaborative waypoint planning network (CWP-net) for trajectory planning in robot joint space. CWP-net is able to learn the DPGMM distribution information of sample paths. Using the starting configuration, the target configuration and the environmental obstacle information as inputs, all the obstacle avoidance waypoints in the joint space are output in a single step. Compared with the widely studied single-step incremental planning methods based on neural networks, CWP-net is able to plan the whole motion trajectory synchronously based on a single output

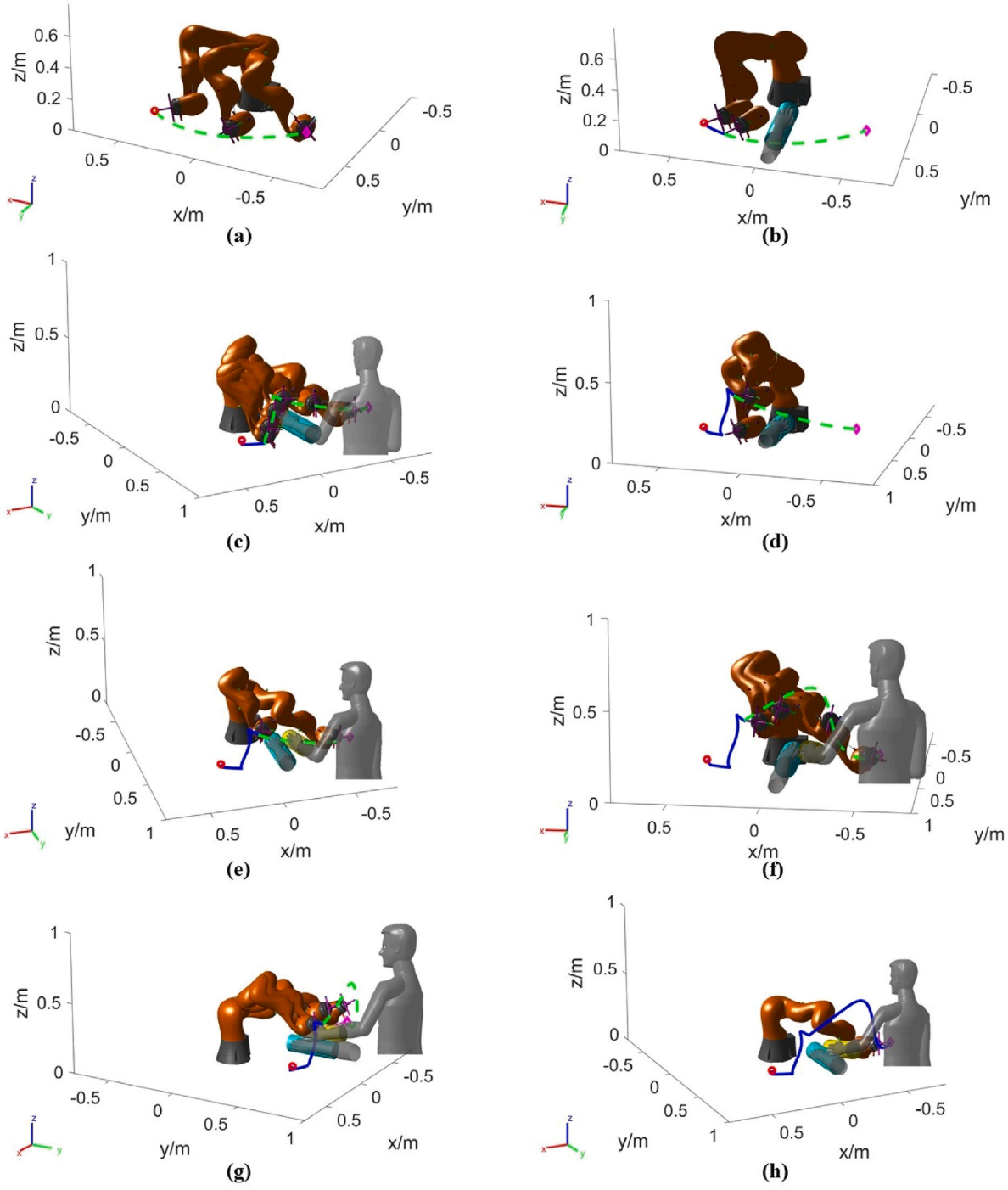



**Fig. 17.** Dynamic simulation with human motion.

based on velocity and acceleration constraints, and is able to combine with other advanced optimization algorithms to control the path cost and motion smoothness. In the second part of the algorithm, we propose an improved STOMP (SWOA) algorithm for locally optimizing the generated trajectories of CWP-net. The method is based on the parameters of the DPGMM distribution model established in the first part, which constrains the random noise generation range and the trajectory optimization direction. While ensuring the optimization quality, it further enhances the planning efficiency. Simulations and real experiments verify the feasibility of the online collision-free trajectory planning algorithm proposed in this paper. Compared with the current state-of-the-art trajectory planning algorithms, the proposed algorithm is 4.0 times and 15.5 times faster than RRT-connect and RRT*, respectively. Also the improved STOMP combined with CWP-net algorithm is at least 2.5 times faster than the original STOMP algorithm. In summary, the

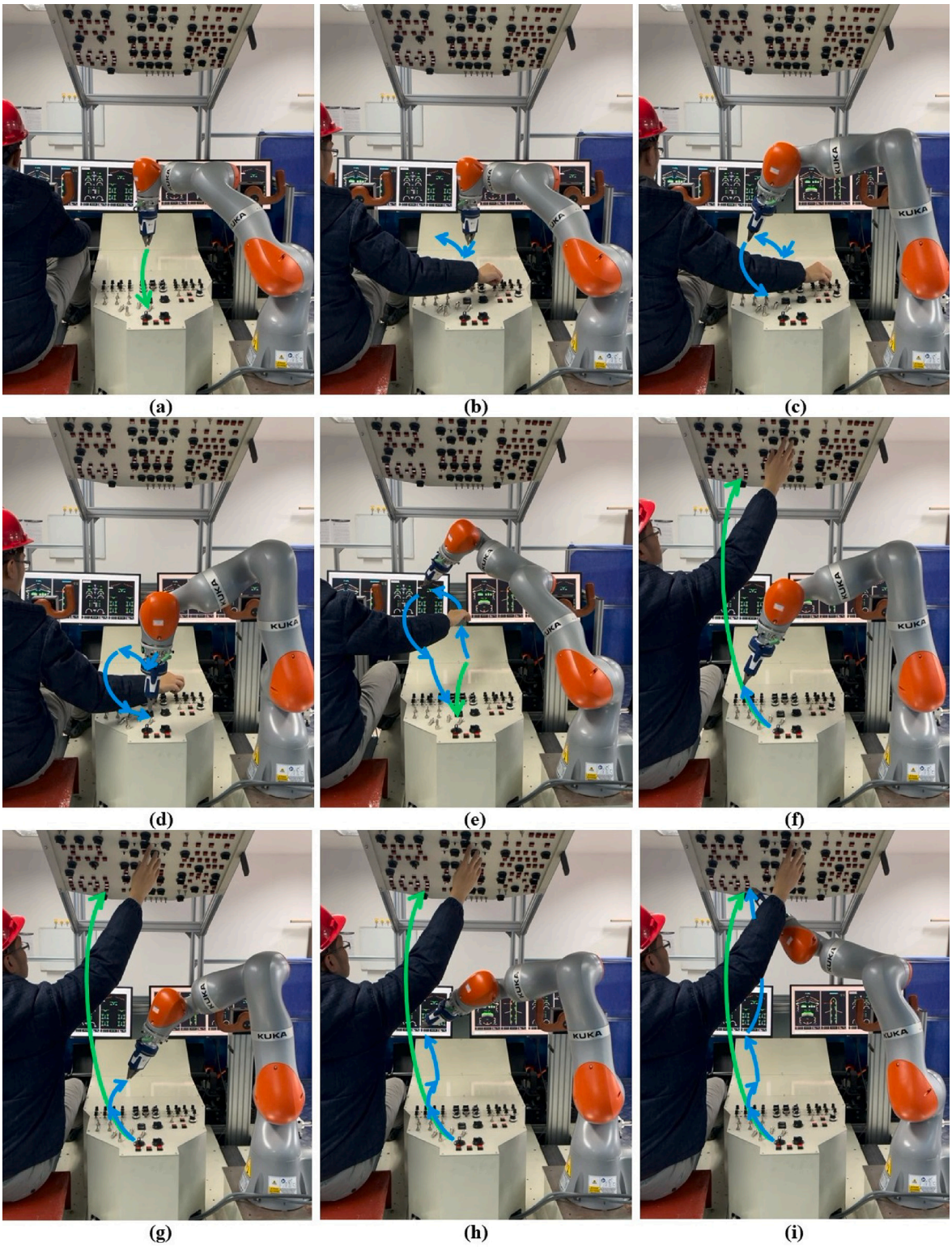



**Fig. 18.** Real collaboration scenario testing with on-line safety trajectory generation.

proposed framework of our algorithm builds a bridge between offline planners and online planning tasks, which can combine the results of advanced planners that do not meet real-time requirements into dynamic tasks of human–robot collaboration type.

The disadvantage of the method in this paper is that the collection of training data is time-consuming. It requires historical data or a classical planner to generate feasible trajectories offline. However, the process only needs to be performed before industrial applications, and no extra operations are required during the work. In addition, although the CWP-net-based neural network planner has a good computational efficiency of 10 ms, which is suitable for a general dynamic HRC scenario. However, the STOMP-based trajectory optimization method temporarily limits HRC with moderately restricted relative motion velocities for application. Parallel versions of the program [15,16] will be developed to utilize multi-core CPUs or multi-core GPUs for parallel

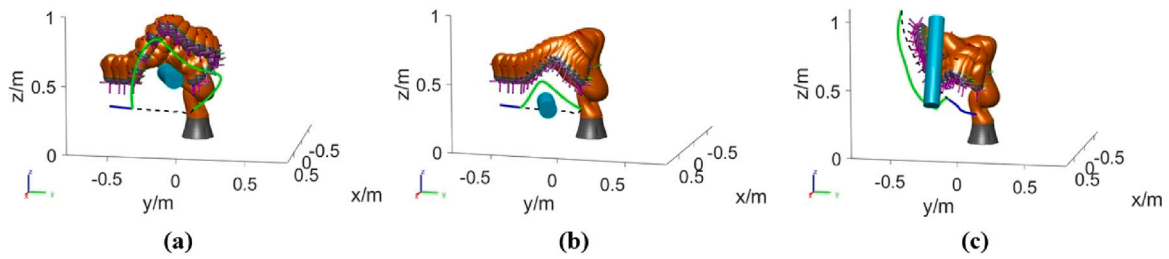



**Fig. 19.** Trajectories generated from human arm obstacles in real collaborative experiments.

computation to improve the SWOA algorithm's efficiency, making the algorithm capable of more dynamic and variant situations.

## CRediT authorship contribution statement

**Yanzhe Wang:** Conceptualization, Methodology, Software, Writing – original draft. **Lai Wei:** Software, Investigation, Formal analysis, Writing – review & editing. **Kunpeng Du:** Resources, Data curation. **Gongping Liu:** Resources, Visualization. **Qian Yang:** Writing – review & editing, Methodology, Software. **Yanding Wei:** Project administration, Supervision, Conceptualization, Resources. **Qiang Fang:** Funding acquisition, Supervision, Conceptualization, Resources.

## Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

## Data availability

The authors do not have permission to share data.

## Acknowledgment

This work is supported by the National Key Research and Development Program of China (2019YFB1707505).

## Appendix A. Empirical formula for workspace discretization

The arm envelope cylinders are located in the voxel-represented workspace, occupying different voxels depending on the different voxel resolution, envelope lengths, radius, safety distances, and occupation rules. Therefore, we can only give a general applicable discrete resolution reference formula based on the specific conditions. However, it is possible to regain the empirical formula applicable to the new conditions by re-following the method described below when some conditions change.

Since the position and pose of the arm cylindrical envelope in the workspace are random, using the Monte Carlo method to obtain the empirical formula is achievable. Still, it will turn out to be extremely costly. Therefore, we use the uniform sampling method to sample the position and pose of the arm cylindrical envelope in the workspace, while making the empirical formula as accurate as possible. The detailed implementation process is as follows.

The arm enveloping cylinder has a radius of 0.06 m and a length of 0.5 m. Initially, the cylinder's main axis is parallel to the $z$-axis in the workspace. The cylinder pose is controlled by rotating the initial pose around the $x$-axis in the workspace by an angle $\alpha$ and then rotating it around the $z$-axis in the workspace by $\beta$, where $\alpha \in \{0, 12°, 24°, \ldots, 348°\}$ and $\beta \in \{0, 12°, 24°, \ldots, 348°\}$. The cylinder position is obtained by translation along the x/y/z axis in the workspace, and the translation step length is defined as $D/8L$, where the $L$ is the workspace edge length and $D$ is the discrete resolution. The total

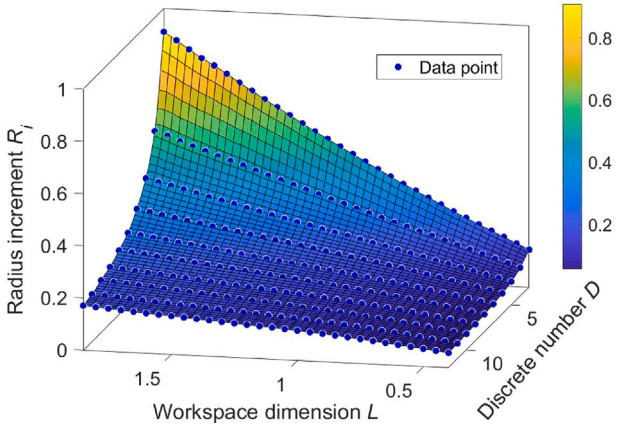



**Fig. A.1.** Accuracy losses in the modeling of the arm cylindrical envelope with its actual volume at different shared workspace sizes and discrete resolutions. (The blue data points are the corresponding results obtained from the simulation. The colored surface is obtained by interpolating all data points). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

number of homogeneous translations is 512. Therefore, for each given $L$ and $D$, 460800 workspace occupancy cases are calculated. The radius increment on the true arm envelope is calculated based on the total volume of the grids occupied by the discrete cylindrical envelope. Finally, the average of all radius increments under this condition is recorded as the incremental radius under the given $L$ and $D$. In our simulation, $L$ is uniformly sampled in steps of 0.05 m for a total of 33 cases from 0.4 m to 2 m. The discrete resolution $D$ is uniformly set from 3 to 12 for 10 resolutions. The results are obtained from the above conditions, as shown in Fig. A.1.

Based on the simulation results, we fit the data with a polynomial function $D(R_i, L) : \{R_i, L\} \in \mathbb{R} \times \mathbb{R} \rightarrow D \in \mathbb{R}$, by the least absolute residual method. The formula is shown in Eq. (1).

## Appendix B. Techniques in data augmentation

Due to the extensive distribution of robot and human collaborator arms in the shared workspace, we use a recursive strategy to generate the required samples of obstacle avoidance motion trajectories to cover as much work task space as possible. According to the starting configuration and target configuration set by the task requirements, we first plan the long-distance initial collision-free path in the case of random obstacle distribution. For the key obstacle avoidance waypoints of each path generated in this round, the configurations which are closer to the starting configuration are selected as the new starting points of the collision-free paths planned to the original targets in the new round. As a result, 1209 sample paths are generated and used to build the DPGMM model that characterizes the historical path point distribution information. These paths are able to cover the various critical obstacle avoidance configurations needed to perform tasks in a shared workspace, while these configurations can also serve as new

starting points for replanning when the original path is infeasible for new obstacles.

It is obvious that one collision-free path is not only suitable for one specific obstacle pose, but is also feasible and optimal for some other similar obstacles. Therefore, we propose some data augmentation techniques on our previously obtained 1209 collision-free paths.

(1) The obstacle translates along the x/y/z axis in some range.

(2) The obstacle rotates along the x/y/z axis in some range.

(3) The obstacle voxel randomly jumps to one of its 26 neighbor voxels.

After generating the augmented data, we filter the data that no collision occurs between the path and the new obstacle configuration and add them into the training dataset. By this method, we randomly expand the original sample of 1209 paths to 6045 path and use them to train the neural network.

## Appendix C. Training and validation of the CWP-net

The traditional technique for training and testing models is to split the data into two different splits, with a typical ratio of 7:3. However, this approach brings problems that we need to manually tune the hyperparameters to achieve a good performance. This static approach leads to the risk of overfitting on the testing set. The evaluation metrics do not reflect the model's generalization performance.

We can further split data into training, validation, and testing sets to solve the above problem. The model optimal hyperparameters are obtained from the training and validation sets, and finally, the model generalization performance is evaluated by the testing set. However, this technique has two disadvantages. The first is that splitting the data into three sets reduces the number of samples used to learn the model's weight parameters, which is very unfriendly for small training datasets, as in our case. The second is that the testing set may not fully represent the distribution of the whole data, leading to a non-optimal training process.

To overcome the problems mentioned above, we use two times of K-fold cross-validation technique to select the model hyperparameter and evaluate the model performance in the training process. The detail of the technique is shown below.

**Step 1.** Randomly split the whole dataset into $k_1 = 5$ folds, namely, $\{S_1, S_2, \ldots, S_5\}$, where $S_5$ is the testing set. This split is known as the outer fold.

**Step 2.** Randomly split the remaining sets $\{S_1, S_2, \ldots, S_4\}$ into $k_2 = 10$ folds, namely, $\{s_1, s_2, \ldots, s_{10}\}$. This split is known as the inner fold. Use $s_1$ as the validation set and the rest merged as the training set. Record the best MSE Loss of the validation set under the given hyperparameter.

**Step 3.** Repeat step 2 for $k_2 = 10$ times. For each time, use one different fold in $\{s_1, s_2, \ldots, s_{10}\}$ as the validation set and the rest as the training set. Calculate the mean and standard deviation as the performance measurement for the current hyperparameter.

**Step 4.** For each different group of hyperparameters, repeat the step 2 and 3.

**Step 5.** Select the best hyperparameter by the mean and standard deviation of each hyperparameter group.

**Step 6.** Use the best hyperparameter selected from step 5, train the model on the combined dataset $\{S_1, S_2, \ldots, S_4\}$, and test the performance on the set $S_5$.

**Step 7.** Repeat step 6, use $s_1$, $s_2$, $s_3$, and $s_4$ as the test set, respectively. Then use the remaining sets for training the model. Select the best among five models as the CWP-net in the real application.

The training process uses the Adam optimizer, with the recommended hyperparameters weight decay of 0.0005, initial learning rate from 0.001 to 0.01, and a dropout rate from 0.2 to 0.5. Consider the polynomial computation complexity of K-fold cross-validation, we search initial learning rate in 0.001, 0.004, 0.007, 0.01 and dropout rate in 0.2, 0.3, 0.4, 0.5. Meanwhile, we search the batch size in 32, 64, 128, and 256. Finally, we get a total of 64 groups of hyperparameters searching for the best hyperparameters. The result of the experiment is shown in Fig. C.1.

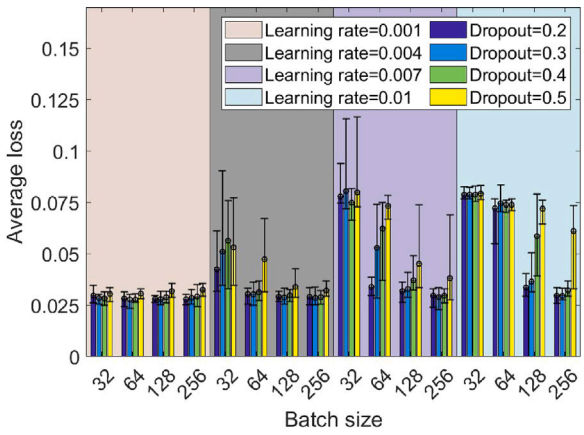



**Fig. C.1.** Inner K-fold cross-validation for hyperparameter group selection with $K = 10$.

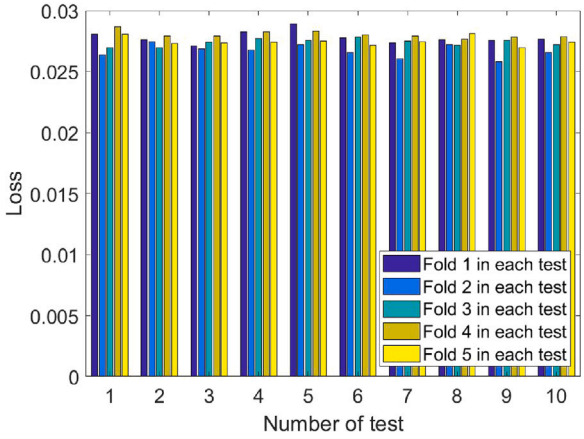



**Fig. C.2.** Outer K-fold cross-validation repeated experiments for optimal model selection with $K = 5$.

Among all hyperparameter groups, we finally select the group of learning rate = 0.001, batch size = 64, and dropout rate = 0.3. Using this setting, we randomly re-split and repeat the outer fold cross validation ten times and obtained the results on each test dataset in Fig. C.2.

The results of the mean and standard deviation of the loss for each five-fold test in ten replicate experiments are shown in Table C.1. The best mean value is 0.0271, and the best standard deviation is 0.000361, corresponding to the 9th and 2nd group of experiments, respectively. However, the larger standard deviation results on the 9th group compared to the other groups indicate that this model suffers from a certain degree of imbalance sample distribution. Also, the generalization performance is not the best among all. We need to filter out the models with good mean and standard deviation performance and finally select 3rd group as the optimal model.

In addition, the results from this cross-validation experiments show that the loss distribution values of the ten five-fold test results are relatively uniform, so the average generalization error of the model on this test set is low. It also shows that there is no overfitting caused by the unreasonable division of the dataset during the model training, which ensures the generalization of the model trained based on this dataset.

At the same time, based on the fact that the overall test results were evenly distributed, we find that the dataset we used for training and

**Table C.1**
Results of repeated five-fold cross-validation tests with 10 random splits.

| Test No. | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| Mean value ($\times 10^{-2}$) | 2.76 | 2.74 | 2.73 | 2.77 | 2.79 | 2.75 | 2.73 | 2.76 | 2.71 | 2.73 |
| Standard deviation ($\times 10^{-4}$) | 9.39 | 3.61 | 3.81 | 6.39 | 7.00 | 6.12 | 7.12 | 3.86 | 8.11 | 5.01 |

testing did not have a large test discrepancy in the ten five-fold cross-validations. The experiment shows our small dataset is well enough to obtain reliable training results. Therefore, this experiment can also demonstrate that the size and the distribution of the dataset for this task are relatively reasonable and can guarantee the model's generalization performance.

## Appendix D. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.rcim.2022.102475.

## References


[1] J. Saenz, N. Elkmann, O. Gibaru, P. Neto, Survey of methods for design of collaborative robotics applications- Why safety is a barrier to more widespread robotics uptake, in: Proceedings of the 2018 4th International Conference on Mechatronics and Robotics Engineering, 2018.

[2] L.E. Kavraki, J.-C. Latombe, Probabilistic Roadmaps for Robot Path Planning, 1998.

[3] S.M. LaValle, Rapidly-exploring random trees : A new tool for path planning, Annu. Res. Rep. (1998).

[4] S. Karaman, E. Frazzoli, Sampling-based algorithms for optimal motion planning, Int. J. Robot. Res. 30 (2011) 846–894.

[5] J.D. Gammell, S.S. Srinivasa, T.D. Barfoot, Informed RRT*: Optimal sampling-based path planning focused via direct sampling of an admissible ellipsoidal heuristic, in: 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2014, pp. 2997–3004.

[6] J.J. Kuffner, S.M. LaValle, RRT-connect: An efficient approach to single-query path planning, in: Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065), Vol. 2, 2000, pp. 995–1001.

[7] N.D. Ratliff, M. Zucker, J.A. Bagnell, S.S. Srinivasa, CHOMP: Gradient optimization techniques for efficient motion planning, in: 2009 IEEE International Conference on Robotics and Automation, 2009, pp. 489–494.

[8] M. Kalakrishnan, S. Chitta, E.A. Theodorou, P. Pastor, S. Schaal, STOMP: Stochastic trajectory optimization for motion planning, in: 2011 IEEE International Conference on Robotics and Automation, 2011, pp. 4569–4574.

[9] J.-J. Kim, J. Lee, Trajectory optimization with particle swarm optimization for manipulator motion planning, IEEE Trans. Ind. Inf. 11 (2015) 620–631.

[10] E.K. Xidias, Time-optimal trajectory planning for hyper-redundant manipulators in 3D workspaces, Robot. Comput.-Integr. Manuf. 50 (2018) 286–298.

[11] D. Chen, S. Li, J. Wang, Y. Feng, Y. Liu, A multi-objective trajectory planning method based on the improved immune clonal selection algorithm, Robotics Comput. Integr. Manuf. 59 (2019) 431–442.

[12] O. Khatib, Real-time obstacle avoidance for manipulators and mobile robots, in: Autonomous Robot Vehicles, 1990.

[13] G. Liu, H. He, G. Tian, J. Zhang, Z. Ji, Online collision avoidance for human-robot collaborative interaction concerning safety and efficiency, in: 2020 IEEE/ASME International Conference on Advanced Intelligent Mechatronics, AIM, 2020, pp. 1667–1672.

[14] M. Safeea, P. Neto, R. Béarée, On-line collision avoidance for collaborative robot manipulators by adjusting off-line generated paths: An industrial use case, 2019, arXiv, arXiv:1909.05159.

[15] C. Park, J. Pan, D. Manocha, ITOMP: Incremental trajectory optimization for real-time replanning in dynamic environments, in: ICAPS, 2012.

[16] O.S. Oguz, O.C. Sari, K.H. Dinh, D. Wollherr, Progressive stochastic motion planning for human-robot interaction, in: 2017 26th IEEE International Symposium on Robot and Human Interactive Communication, RO-MAN, 2017, pp. 1194–1201.

[17] J. Kim, E.A. Croft, Online near time-optimal trajectory planning for industrial robots, Robotics Comput. Integr. Manuf. 58 (2019) 158–171.

[18] H. Simas, R.D. Gregorio, Smooth transition for collision avoidance of redundant robots: An on-line polynomial approach, Robotics Comput. Integr. Manuf. 72 (2021) 102087.

[19] J. Mainprice, R. Hayne, D. Berenson, Goal set inverse optimal control and iterative replanning for predicting human reaching motions in shared workspaces, IEEE Trans. Robot. 32 (2016) 897–908.

[20] R. Meziane, M.J.-D. Otis, H. Ezzaidi, Human-robot collaboration while sharing production activities in dynamic environment: SPADER system, Robot. Comput.-Integr. Manuf. 48 (2017) 243–253.

[21] A.H. Qureshi, Y. Miao, A. Simeonov, M.C. Yip, Motion planning networks: Bridging the gap between learning-based and classical motion planners, IEEE Trans. Robot. 37 (2021) 48–66.

[22] M.J. Bency, A.H. Qureshi, M.C. Yip, Neural path planning: Fixed time, near-optimal path generation via oracle imitation, in: 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS, 2019, pp. 3965–3972.

[23] B. Ichter, J. Harrison, M. Pavone, Learning sampling distributions for robot motion planning, in: 2018 IEEE International Conference on Robotics and Automation, ICRA, 2018, pp. 7087–7094.

[24] M. Duguleană, F. Barbuceanu, A. Teirelbar, G. Mogan, Obstacle avoidance of redundant manipulators using neural networks based reinforcement learning, Robot. Comput.-Integr. Manuf. 28 (2012) 132–146.

[25] C. Wurll, D. Henrich, Point-to-point and multi-goal path planning for industrial robots, J. Field Robot. 18 (2001) 445–461.

[26] C.E. Rasmussen, The infinite Gaussian mixture model, in: NIPS, 1999.

[27] D.H. Douglas, T.K. Peucker, Algorithms for the reduction of the number of points required to represent a digitized line or its caricature, Cartographica: Int. J. Geogr. Inform. Geovisual. 10 (1973) 112–122.

[28] A.P. Dempster, N.M. Laird, D.B. Rubin, Maximum likelihood from incomplete data via the EM - Algorithm plus discussions on the paper, 1977.

[29] K. Shibata, Y. Ikeda, Effect of number of hidden neurons on learning in large-scale layered neural networks, in: 2009 ICCAS-SICE, 2009, pp. 5008–5013.

[30] K. Qin, General matrix representations for B-splines, Vis. Comput. 16 (2014) 177–186.

[31] Matlab, Curve fitting toolbox, https://ww2.mathworks.cn/products/curvefitting.html.

[32] Z. Zhang, L. Zheng, Z. Chen, L. Kong, H.R. Karimi, Mutual-collision-avoidance scheme synthesized by neural networks for dual redundant robot manipulators executing cooperative tasks, IEEE Trans. Neural Netw. Learn. Syst. 32 (2021) 1052–1066.

[33] F. Chollet, Keras, 2015, https://keras.io/.

[34] I.A. Şucan, M. Moll, L.E. Kavraki, The open motion planning library, IEEE Robot. Autom. Mag. 19 (4) (2012) 72–82, http://dx.doi.org/10.1109/MRA.2012.2205651.

[35] I.A.C. Sucan, MoveIt, https://moveit.ros.org/.

[36] F. Zhang, V. Bazarevsky, A. Vakunov, A. Tkachenka, G. Sung, C.-L. Chang, M. Grundmann, MediaPipe hands: On-device real-time hand tracking, 2020, arXiv, arXiv:2006.10214.