Measurement Science and
Technology

PAPER • OPEN ACCESS

An effective self-collision detection algorithm for multi-degree-of-freedom manipulator

To cite this article: Zhenyu Liu *et al* 2023 *Meas. Sci. Technol.* **34** 015901

OPEN ACCESS
IOP Publishing

Measurement Science and Technology

Meas. Sci. Technol. **34** (2023) 015901 (11pp)

https://doi.org/10.1088/1361-6501/ac9920

An effective self-collision detection algorithm for multi-degree-of-freedom manipulator

Zhenyu Liu[1], Lianhao Zhang[1,*], Xiaohong Qin[2] and Gang Li[3]

[1] School of Information Science and Engineering, Shenyang University of Technology, Shenyang, People's Republic of China
[2] Department of Information and Control Engineering, Shenyang Institute of Science and Technology, Shenyang, People's Republic of China
[3] Shenyang SIASUN Robot & Automation Co., Ltd, Shenyang, People's Republic of China

E-mail: xxx_ace@163.com

Received 12 June 2022, revised 25 September 2022
Accepted for publication 11 October 2022
Published 19 October 2022

Abstract

In order to address the self-collision problem associated with the operation of modern industrial robots, this paper proposes a multi-degree-of-freedom collision detection algorithm that can detect self-collision in single arm and double arm robots as well as collision with the load. Firstly, the zero pose of the Denaviti−Hartenberg model is built based on the manipulator configuration, and the coordinate information of each key point is obtained through a rotation and translation operation of the matrix. Then, the positional relation and distance between the detected objects are determined by the spatial geometry theory, and finally, collision is detected using a collision matrix. By simulating two groups of single arms and two groups of double arms, and from the laboratory testing of SR10C in the SIASUN robot factory, it has been verified that the proposed algorithm has good collision detection capability. Without the use of sensors, cameras, and other external devices, the collision between the arm and the load, and the collision between the cooperative robot and the load may be effectively detected and mitigated.

Keywords: multi-DOF, self-collision, distance detection, manipulator, load

(Some figures may appear in colour only in the online journal)

1. Introduction

As a result of the rapid development of robotic technologies, the demand for more intelligent algorithms is increasing. Accordingly, the self-collision problem of multi-degree-of-freedom (multi-DOF) manipulators for intelligent path planning has been at the forefront of robotics research.

Himmelsbach *et al* [1] used a single pixel time-of-flight sensor to detect obstacles to the robot arm and thereby prevent collision. The inertial measurement unit sensor was used to measure the joint angle of the manipulator and monitor collision [2, 3]. In Liang *et al* [4, 5] the detection information is fed back through the collision surface of the tactile sensor. At present, the effect of sensor application is better, but it will increase the load on the manipulator and the cost.

The KUKA robot arm used a deformeter to measure the external torque in each joint for collision detection. Zhang *et al* [6] designed a pair of envelope lines that can detect the collision of industrial robots. Using the joint structure, Alberto *et al* [7] proposed an ontology-aware collision detection algorithm based on Gaussian regression. Most of the

[*] Author to whom any correspondence should be addressed.

motor measurement or deformeter based methods are associated with post-collision detection, and the detection accuracy of the torque difference will decrease with the variation in time. Moreover, the wiring of the skin perception is complex and the anti-interference performance is poor.

Several researchers opted to discretize the operation space into small discrete volumes, and used the master-slave and occupied-free models, along with other methods to work [8, 9]. Zhou *et al* [10] proposed a collision-free compliance control strategy based on the physical constraints. Sangiovanni *et al* [11] used deep reinforcement learning to develop a hybrid control method to avoid robot body collision. Park *et al* [12] designed a detection method based on a learning support vector machine and convolutional neural networks. In order to avoid the collision between the ball shoulder and wrist joint of the robot and the body, Oh *et al* [13] used the arm angle to mitigate the redundancy of the manipulator. Further, ROS was used to construct a CollisionRequest object and a CollisionResult object to detect collision [14, 15]. Kramar *et al* [16] proposed a method for the external obstacle detection by a dual-arm robot. Division of space methods usually involves extensive computation, and the method of joint angle limitation reduces the working range of the manipulator and reverse solution rate. Other intelligent algorithms have long detection time and poor stability, which are not advisable for application in the case of industrial manipulators.

This paper puts forward a simple and effective fast collision detection algorithm based on mathematical model analysis for actual engineering needs. When the state of collision is reached, it will sound an alarm and halt the operation of the manipulator. The essential requirements of the algorithm come from the project requirement of the SIASUN Robot company. The 6-DOF SR10C manipulator is used to carry liquid crystal display and other loads, and the 4-DOF manipulator is used to cooperate. The conventional method is to use the teaching mode, but when the size of the load or the position and posture of the manipulator are changed, collision and subsequent damage to the object or the manipulator itself may occur. The objective of the algorithm examined in this paper is to solve the self-collision problem, and the simulation and self-collision test of different types of manipulators are carried out on this basis. This is expected to provide more in-depth knowledge of the usability of the algorithm.

This paper is structured in the following manner. In section 2, the coordinate information of the key points are obtained. In section 3, we describe the basic principle and provide the flowchart of the algorithm, while in section 4 the experiment and analysis of the collision detection process are carried out using different types of manipulators. Finally, section 5 provides a brief conclusion.

2. Get the coordinate information of key points

It is necessary to determine the key points of the detected object and know the spatial position information before the algorithm can be applied, and most of these key points are joint points or their offsets. For the joint limit constraints and collision-free constraints for hyper-redundant manipulators, Zhao *et al* [17] provide an algorithm based on the relation between the forward and backward inverse kinematics. A manipulator is a complex chain structure composed of a series of links. Denavit and Hartenberg proposed a matrix parameter analysis method (DH parameter), which established a 4 × 4 homogeneous transformation matrix for the links of each joint, and expressed the relationship between each link using θ_i, d_i, a_i and α_i. According to matrix transformation, the coordinates of each joint were unified in the base coordinate system, which made preparations the for collision detection.

In order to realize the coordinate transformation, a space coordinate system *x, y,* and *z* are set for each joint. Among these, θ_i is the angle of the common perpendicular line between two links from x_{i-1} to the x_i axis around the z_{ii-1} axis; a_i is the distance between the common normals of the two links from the z_{ii-1} to z_i axis along the x_i axis; α_i is the angle from z_{ii-1} to z_i axis around x_i axis in the plane vertical to a_i; d_i is relative distance between the two links from the x_{ii-1} to x_i axis along the z_i axis.

The homogeneous transformation of the relative translation and rotation between the link coordinate systems is described by the A_i matrix, which can be written as:

$$A_i = Rot(z,\theta_i)Trans(0,0,d_i)Trans(a_i,0,0)Rot(x,\alpha_i). \quad (1)$$

The A_{i} matrix can be computed efficiently according to:

$$Rot(z,\theta_i) = \begin{bmatrix} \mathrm{c}\theta_i & -\mathrm{s}\theta_i & 0 & 0 \\ \mathrm{s}\theta_i & \mathrm{c}\theta_i & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad (2)$$

$$Trans(0,0,d_i) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad (3)$$

$$Trans(a_i,0,0) = \begin{bmatrix} 1 & 0 & 0 & a_i \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad (4)$$

$$Rot(\mathrm{x},\alpha_i) = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & \mathrm{c}\alpha_i & -\mathrm{s}\alpha_i & 0 \\ 0 & \mathrm{s}\alpha_i & \mathrm{c}\alpha_i & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \quad (5)$$

Through matrix (2)–(5), we get:

$$A_i = \begin{bmatrix} \mathrm{c}\theta_i & -\mathrm{s}\theta_i\mathrm{c}\alpha_i & \mathrm{s}\theta_i\mathrm{s}\alpha_i & a_i\mathrm{c}\theta_i \\ \mathrm{s}\theta_i & \mathrm{c}\theta_i\mathrm{c}\alpha_i & -\mathrm{c}\theta_i\mathrm{s}\alpha_i & a_i\mathrm{s}\theta_i \\ 0 & \mathrm{s}\alpha_i & \mathrm{c}\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}. \quad (6)$$

The *s* and *c* in the matrix represent the sine and cosine operations respectively. Based on the values of θ_i, d_i, a_i, and α_i for each joint in the DH parameter table, A_1, A_2, A_3, A_4, A_5 and A_6

can be obtained one by one, and the coordinate information of the end-effort for 6-DOF is:

$$T6 = A_1A_2A_3A_4A_5A_6 = \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix}. \quad (7)$$

In the same way, $T5 = A_1A_2A_3A_4A_5$, $T4 = A_1A_2A_3A_4$, $T3 = A_1A_2A_3$, $T2 = A_1A_2$, $T1 = A_1$, the values of the spatial key points of other joints are obtained, and these coordinate values are relative to the same coordinate system (base coordinate system). The additional key points can be used in the DH zero pose parameter modeling, or through the joint value matrix rotation, translation, and other operations to obtain the corresponding coordinate information.

On the other hand, if there is a tool at the end of the manipulator to carry out the load operation, an outer cuboid or sphere is established with the center of mass of the load as the center, and the key points of each edge and the center of the sphere are determined, which can be obtained by the offset of the pose of the end-effort.

3. Get the coordinate information of key points

While designing algorithms for collision detection and obstacle avoidance, many experts simplify the target into a topological structure. Some studies made an effective bounding box modeling of obstacles, and the collision detection was realized by using the vector relationship between them [18, 19]. In this work, the idea of simplification is used for reference, and each link of the multi-DOF manipulator is surrounded by a cylinder. On the basis of obtaining the coordinate information of each joint, it is simplified into the comparison and determination of the topological line of the cylinder axis, and the corresponding collision detection is completed. The basic principle and process of algorithm application mainly include determining the coordinates of the key points, cylinder link positions, and collision distance detection during traversal.

Initializing the pose and updating the coordinate values of the key points is the first task to be completed. The coordinate values of each joint point are obtained through the parameter modeling of the manipulator. Due to the configuration characteristics of the manipulator, the coordinate value of the joint point should be converted into the coordinate value of the link and expressed as the coordinate value of the key point as far as possible, so as to reduce the need for computation. When the manipulator is in use, the coordinates of the key points are updated from the kinematic relations.

Next, it is necessary to determine the presentation form of the link and form a collision detection matrix. Each link is simplified into the form of a cylinder, and the coordinates at both ends of the central axis of the cylinder are the coordinates of the key points. The problem of self-collision can be simplified as the determination of the position relationship of the central axis of any two links. With the working beat of the manipulator, set the detection time of the interval to judge whether link of the manipulator has self-collision.

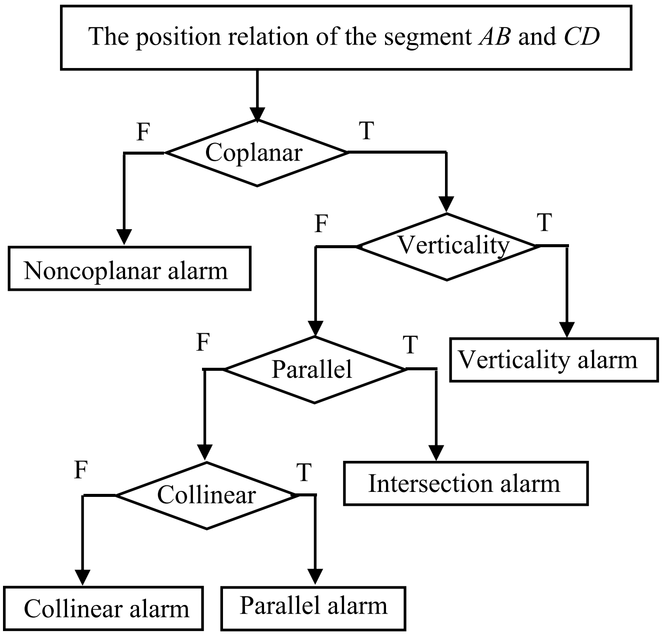


Figure 1. Flowchart of collision detection for two links.

3.1. Flowchart

Taking two link cylinders as an example, the position relationship of the central axis of the link is determined, and the self-collision detection and alarm generation are completed as shown in the flowchart in figure 1.

Consider the endpoint coordinates of the two line segments to be *A(Xa,Ya,Za)*, *B(Xb,Yb,Zb)*, *C(Xc,Yc,Zc)*, and *D(Xd,Yd,Zd)*, and the cylinder radius of the links are $r1$ and $r2$. According to the flowchart, the two line segments at the center, *AB* and *CD*, are examined successively to check whether the spatial position relations such as coplanarity, verticality, parallelism, and collinearity, and the corresponding distance detection and alarm function were carried out.

3.2. Judgment of spatial position relationship

3.2.1. Coplanar relation. According to the coordinates of *A*, *B* and *C*, we can get the equation:

$$\begin{aligned} a &= Ya * (Zb - Zc) + Yb * (Zc - Za) + Yc * (Za - Zb) \\ b &= Za * (Xb - Xc) + Zb * (Xc - Xa) + Zc * (Xa - Xb) \\ c &= Xa * (Yb - Yc) + Xb * (Yc - Ya) + Xc * (Ya - Yb) \\ d &= -Xa * (Yb * Zc - Yc * Zb) + Xb * (Yc * Za \\ &\quad - Ya * Zc) + Xc * (Ya * Zb - Yb * Za). \end{aligned} \quad (8)$$

And, the standard form of the coplanar equation of the three points is:

$$ax + by + cz + d = 0. \quad (9)$$

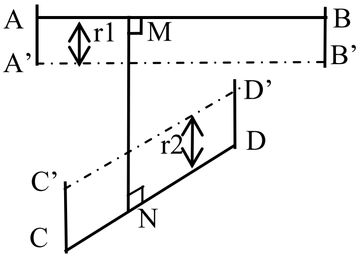


Figure 2. Noncoplanar line distance detection.

Then, judge whether point D satisfies equation (9), if the condition is true, then points A, B, C, and D are coplanar.

3.2.2. Verticality relation. When the vectors satisfy $\overrightarrow{AB}*\overrightarrow{CD}=0$, then the line segments AB and CD are vertical.

3.2.3. Parallel relation. The vectors $\overrightarrow{AB}$ and $\overrightarrow{CD}$ can be obtained based on the coordinates of A, B, C, and D. If it exists as $\overrightarrow{AB}=\lambda\overrightarrow{CD}$, the line segments AB and CD are parallel; otherwise they intersect in the same plane.

3.2.4. Collinear relation. Euclidean geometry is used to estimate the distance of each point, and the areas S_{ABC} and S_{ABD} are obtained. If the areas reduce to zero simultaneously, then the four points are collinear.

3.3. Collision distance detection

3.3.1. Noncoplanar distance detection. Determination of the noncoplanar situation is the most complex part of implementing the algorithm. This includes considering not only the distance between each other, but also obtaining the coordinates of the common perpendicular foot of the two central axes. We can obtain the spatial coordinates of the four points based on the method mentioned before, and the segments AB and CD are noncoplanar. If AB and CD have a common perpendicular line, the points $M(Xm,Ym,Zm)$ and $N(Xn,Yn,Zn)$ may be set as the feet of the perpendiculars of AB and CD. When the coordinates of the two common perpendicular foot points are obtained, the distance between the two planes can be calculated and the alarm given as shown in figure 2.

According to the relation of vectors $\overrightarrow{AM}=t1*\overrightarrow{AB}$, we obtain the spatial coordinates as follows:

$$M\begin{pmatrix} t1\,(Xb-Xa)+Xa, \\ t1\,(Yb-Ya)+Ya, \\ t1\,(Zb-Za)+Za \end{pmatrix}, N\begin{pmatrix} t2\,(Xd-Xc)+Xc, \\ t2\,(Yd-Yc)+Yc, \\ t2\,(Zd-Zc)+Zc \end{pmatrix}. \tag{10}$$

Then, the coordinates M and N can be obtained from the values of $t1$ and $t2$. The vector is:

$$\begin{aligned}\overrightarrow{MN} = (&t2\,(Xd-Xc)+Xc-t1\,(Xb-Xa)-Xa,\\ &t2\,(Yd-Yc)+Yc-t1\,(Yb-Ya)-Ya,\\ &t2\,(Zd-Zc)+Zc-t1\,(Zb-Za)-Za).\end{aligned} \tag{11}$$

The segment MN is perpendicular to both AB and CD, and can be expressed as:

$$\begin{aligned} &t2\begin{bmatrix}(Xb-Xa)*(Xd-Xc)+(Yb-Ya)\\ *(Yd-Yc)+(Zb-Za)*(Zd-Zc)\end{bmatrix}\\ &-t1\begin{bmatrix}(Xb-Xa)*\ \ (Xb-Xa)+(Yb-Ya)\\ *(Yb-Ya)+(Zb-Za)*(Zb-Za)\end{bmatrix}\\ &+\begin{bmatrix}(Xb-Xa)*(Xc-Xa)+(Yb-Ya)\\ *(Yc-Ya)+(Zb-Za)*(Zc-Za)\end{bmatrix}=0,\end{aligned} \tag{12}$$

$$\begin{aligned} &t2\begin{bmatrix}(Xd-Xc)*(Xd-Xc)+(Yd-Yc)\\ *(Yd-Yc)+(Zd-Zc)*(Zd-Zc)\end{bmatrix}\\ &-t1\begin{bmatrix}(Xb-Xa)*(Xd-Xc)+(Yb-Ya)\\ *(Yd-Yc)+(Zb-Za)*(Zd-Zc)\end{bmatrix}\\ &+\begin{bmatrix}(Xd-Xc)*(Xc-Xa)+(Yd-Yd)\\ *(Yc-Ya)+(Zd-Zc)*(Zc-Za)\end{bmatrix}=0.\end{aligned} \tag{13}$$

From the hypothesis,

$$\begin{aligned} F1\,(a,b) &= \begin{bmatrix}(Xb-Xa)*(Xb-Xa)+(Yb-Ya)\\ *(Yb-Ya)+(Zb-Za)*(Zb-Za)\end{bmatrix}\\ F1\,(c,d) &= \begin{bmatrix}(Xd-Xc)*(Xd-Xc)+(Yd-Yc)\\ *(Yd-Yc)+(Zd-Zc)*(Zd-Zc)\end{bmatrix}\\ F2 &= \begin{bmatrix}(Xb-Xa)*(Xd-Xc)+(Yb-Ya)\\ *(Yd-Yc)+(Zb-Za)*(Zd-Zc)\end{bmatrix}\\ F3\,(a,b) &= \begin{bmatrix}(Xb-Xa)*(Xc-Xa)+(Yb-Ya)\\ *(Yc-Ya)+(Zb-Za)*(Zc-Za)\end{bmatrix}\\ F3\,(c,d) &= \begin{bmatrix}(Xd-Xc)*(Xc-Xa)+(Yd-Yc)\\ *(Yc-Ya)+(Zd-Zc)*(Zc-Za)\end{bmatrix},\end{aligned} \tag{14}$$

Substituting in (12) and (13) and rearranging, we obtain the expressions $t1$ and $t2$. Then, the coordinates of the two perpendicular points are:

$$\begin{aligned} Xm &= t1*(Xb-Xa)+Xa\\ &= (Xb-Xa)*[F3\,(a,b)*F1\,(c,d)-F3\,(c,d)\\ &\quad *F2]/[F1\,(a,b)*F1\,(c,d)-F2*F2]+Xa\\ Ym &= t1*(Yb-Ya)+Ya\\ &= (Yb-Ya)*[F3\,(a,b)*F1\,(c,d)-F3\,(c,d)\\ &\quad *F2]/[F1\,(a,b)*F1\,(c,d)-F2*F2]+Ya\\ Zm &= t1*(Zb-Za)+Za\\ &= (Zb-Za)*[F3\,(a,b)*F1\,(c,d)-F3\,(c,d)\\ &\quad *F2]/[F1\,(a,b)*F1\,(c,d)-F2*F2]+Za,\end{aligned} \tag{15}$$

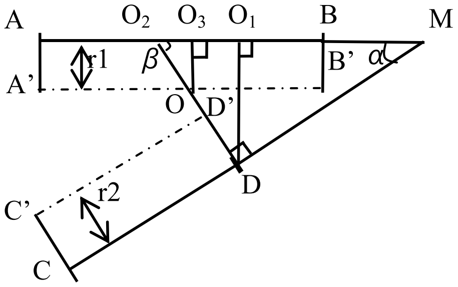


Figure 3. Intersection distance detection.

$$\begin{aligned}Xn &= t2*(Xd-Xc)+Xc\\&=(Xd-Xc)*[F3(c,d)*F1(a,b)-F3(a,b)\\&\quad *F2]/[F2*F2-F1(a,b)*F1(c,d)]+Xc\\Yn &= t2*(Yd-Yc)+Yc\\&=(Yd-Yc)*[F3(c,d)*F1(a,b)-F3(a,b)\\&\quad *F2]/[F2*F2-F1(a,b)*F1(c,d)]+Yc\\Zn &= t2*(Zd-Zc)+Zc\\&=(Zd-Zc)*[F3(c,d)*F1(a,b)-F3(a,b)\\&\quad *F2]/[F2*F2-F1(a,b)*F1(c,d)]+Zc.\end{aligned}\tag{16}$$

The distance between MN is based on equations (15) and (16). It will give a noncoplanar alarm if the length of MN is less than the sum of $r1$ and $r2$, point M is on the line of AB, and N is on the line of CD.

3.3.2. Vertical distance detection. It is relatively simple to judge the vertical condition, only considering the coordinate position of the perpendicular point and the radius of the two cylindrical outer bodies.

3.3.3. Intersection distance detection. For the coplanar intersection of segments AB and CD, three cases may be considered: the intersection point on AB and CD and the extension segment of AB and CD that can be used to determine the distance as shown in figure 3. The A′B′ and C′D′ are the line segments along the outer line of the cylinder where the line segments AB and CD are located respectively.

As long as the intersection occurs along the outer line, it is considered that two links collided. Make a vertical line AB through point D, and the foot point is O_1, the extension of DD' intersects AB at point O_2 and intersects $A'B'$ at point O, and make a vertical line AB through point O, and the foot point is O_3. The α is the angle between the extension lines AB and CD. The distance of DM and the value of angle α can be obtained from the known coordinate information, and then solving to the distance of DO_2, the value of angle of β and the distance of OO_2 based on the trigonometric function operation.

The corresponding value is calculated, and if the distance DO_2 is less than the sum of OO_2 and $r2$, coplanar intersection alarm will occur.

3.3.4. Parallel distance detection. If the line segments AB and CD are parallel, and P and Q are the coordinates of the midpoint of the segments AB and CD, respectively. Then, the length of PQ can be calculated, and point M is the perpendicular foot from the midpoint P of the segment AB to CD. Suppose that the linear equations of AB and CD are:

$$(x-x1)/m=(y-y1)/n=(z-z1)/p,\tag{17}$$

$$(x-x2)/m=(y-y2)/n=(z-z2)/p.\tag{18}$$

The points $M1$ $(x1,\ y1,\ z1)$ and $M2$ $(x2,\ y2,\ z2)$ are on two different line segments, and the direction vector $\overrightarrow{S}=(m,n,p)$. Then,

$$\overrightarrow{M1M2}=(x2-x1,y2-y1,z2-z1)=(a,b,c).\tag{19}$$

The distance PM between parallel lines is obtained:

$$PM=\frac{\sqrt{(bp-cn)^2+(cm-ap)^2+(an-bm)^2}}{(m^2+n^2+p^2)}.\tag{20}$$

When the length of segment MQ is half the sum of AB and CD, then the length of PQ is the square root of the sum of squares of PM and MQ. If the length of PM is less than the sum of $r1$ and $r2$, and the length of PQ is less than the square root of the sum of squares of PM and MQ, the two segments have overlapping areas. This will give an alarm indicating parallelism.

When the length of segment MQ is half the sum of AB and CD, then the length of PQ is the square root of the sum of squares of PM and MQ. If the length of PM is less than the sum of $r1$ and $r2$, and the length of PQ is less than the square root of the sum of squares of PM and MQ, the two segments have overlapping areas. This will give an alarm indicating parallelism.

3.3.5. Collinear distance detection. When the four points A, B, C, and D are collinear, assuming that P and Q are midpoint coordinates of the AB and CD, respectively, the coordinates of P and Q can be obtained, and the length of line segment PQ can be also obtained. If the length of PQ is less than half of the sum of AB and CD, an alarm will be generated to indicate collinearity.

The detection of collision possibility, in the above cases, are not limited to the judgment of distance, but also on detecting whether there are overlapping areas in the collision target. The entire process of collision detection is the self-collision matrix formed by each link, and all non-adjacent links must be so-evaluated. In addition, a global variable *WARN* is set for the application of the algorithm. If any two links collide, the *WARN* is set to 1, a logical value is returned, the operation of the manipulator is stopped immediately, and the corresponding collision information is outputted.

4. Experiment and analysis

In order to verify the effectiveness of the algorithm, the experiment comprises two cases of collision tested for four groups.

Table 1. Zero pose of manipulators.

Group no.	Model	Joint no.	θ (°)	D (m)	a (m)	a (°)	Range (°)	Radius (m)
1	SR5A	1	−90	0.2365	0.0000	0	±175	0.064
		2	−90	0.1500	0.0000	−90	±155	0.064
		3	−90	−0.1500	0.4400	0	±155	0.064
		4	0	0.4100	0.0000	−90	±180	0.061
		5	0	0.1500	0.0000	90	±170	0.061
		6	0	0.1500	0.0000	−90	±180	0.055
2	SR10C	1	0	0.1600	0.0000	−90	±170	0.200
		2	0	0.0000	0.5750	0	−150 ∼ 90	0.100
		3	0	0.1300	0.0000	−90	−82 ∼ 95	0.080
		4	0	0.6450	0.0000	90	±175	0.060
		5	90	0.0000	0.1300	90	±135	0.045
		6	0	−0.1095	0.0000	−90	±360	0.040
3	Double-UR5	1	0	±0.0891 ± 0.2	0.0000	0	±360	0.060
		2	0	0.0000	−0.4250	−90	±360	0.060
		3	0	0.0000	0.3920	0	±360	0.060
		4	0	0.1091	0.0000	0	±360	0.060
		5	0	0.0946	0.0000	90	±360	0.060
		6	0	0.0823	0.0000	90	±360	0.060
4	Zu3	1	−90	0.1506	−0.4000	0	±270	0.055
		2	−90	−0.1150	0.0000	−90	−85 to 265	0.055
		3	0	0.1095	0.2460	0	±175	0.055
		4	90	−0.1175	0.2280	−90	−85 to 265	0.055
		5	0	−0.1175	0.0000	90	±270	0.055
		6	0	−0.1050	0.0000	−90	±270	0.055
	G10	1	0	−0.5000	0.5000	0	±152	0.090
		2	0	0.0000	−0.4000	0	±152	0.4 × 0.04 × 0.02
		3	0	0.0000	0.0000	0	0 to 0.018	0.015
		4	0	0.0000	−0.2500	0	±360°	0.25 × 0.04 × 0.13

Among them, the SR5A and SR10C of SIASUN are used for single arm self-collision detection and load self-collision detection, respectively. The Universal-Robot is used in double arm self-collision detection and load collision detection in the same manner as UR5. The JAKA Zu3 and Epson SCARA G10 are used in double arm self-collision detection and load collision detection of different types. The computer configuration is Intel (R) core (TM) i7-9750 h CPU, @ 2.60 GHz, 16GB RAM with a Windows 10 OS. MATLAB r2018a and Visual C++ 6.0 are used to model, simulate, and test the robots.

Firstly, the DH parameter table of zero pose is established based on the parameters such as link length, radius and joint angle range of the actual manipulator model. At the same time, for collision detection with load, the corresponding shape and size are simulated based on common engineering requirements. Then, the joint values of the manipulator are randomly selected. When the distance between the two detected targets was less than 1 cm, collision alarm occurs. The state with 200 collisions are selected to test the self-collision algorithm, while one group is selected for the experimental analysis and effect explanation. In addition, compared with existing collision detection methods, the proposed algorithm defines the improvement and advantages of this research, and explains the reliability and effectiveness of the work from the perspective of science and technology.

4.1. Establishing DH parameter table of manipulator zero position

The DH parameter established by the conventional method represents the joint coordinate system in the form of simplified data that cannot directly and effectively represent the actual state of the manipulator, especially the multiple right angle joints. In order to verify the intuitiveness and effectiveness of the algorithm, the initial DH parameters are consistent with the zero position state of the manipulator to facilitate obtaining the coordinate information of the key points and the operation, as shown in table 1. At the same time, the coordinate information of the other related offset points can also be obtained quickly, which could save time and calculation cost.

The four groups of manipulators all have six DOFs except G10, and each link is treated as a cylinder. The G10 is a manipulator with four DOFs, the third is a telescopic joint, and the others are rotating joints. Combined with its configuration characteristics, the two links at the second and third parts are constructed by the bounding box of cuboid. The configuration of the SR10C manipulator is relatively complex and the radius scale of each link is quite different, which can minimize the collision error. For other types of manipulators, each link is a cylinder, which is also the constraint that guarantees the high-quality implementation of this algorithm. The simpler the configuration, the smaller the error.

Table 2. Self-collision detection.

Group No.	Model	Joint angle (°)	Link	Key points	x	y	z	d (mm)	Time (ms)
1	SR5A	−90, −124, 64, 0, 104, 0.	L3	P2	0.1300	0.0000	0.2365	5.8	11.82
				P3	0.1300	0.0000	0.6013		
			L7	P6	0.1300	−0.1090	0.3963		
				P7	0.1300	−0.1090	0.3002		
2	SA10C	20.4, −154, 77, −31.5, 135, 0.	L1	P0	0	0	0	8.6	11.94
				P1	0	0	0.1600		
			L6	P5	0.1435	0.1408	0.1699		
				P6	0.0730	0.0714	0.1230		
3	Double-UR5	54, −28.2, 90, 5, 0, 26.2.	R-L6	P5	0.5472	−0.0575	−0.2627	7.1	64.22
				P6	0.5955	−0.0575	−0.1961		
		54, 25.2, −79.2, –46.8, 0.2, 3.7	L-L6	P5	0.5473	0.0600	−0.2628		
				P6	0.5957	0.0600	−0.1962		
4	Zu3-G10	12, −11.1, −59.9, 62, 53, 0.	L6	P6	−0.0658	−0.0542	0.4135	6.3	73.05
				P7	−0.0838	−0.0580	0.5295		
		9.04, −68, 0.008, 1.8	CD	C	−0.0582	−0.1307	0.4900		
				D	0.0707	0.0835	0.4900		

For the Groups 3 and 4, the offset values of some positions are added. The value of double-UR5 is 0.0891 + 0.2 and −0.0891 − 0.2 respectively, the value of JAKA Zu3 is −0.4, and SCARA G10 is −0.5 and 0.5. Through this setting, the actual working environment of the manipulator is met, and the verification of the algorithm is more in line with scene application.

4.2. Self-collision detection

For the common engineering assembly operation, the manipulator mostly adopts the traditional teaching mode. The trajectory of each link and joint is specified in advance, and there is no self-collision for the manipulator bodies. However, with the needs of different industries, this simple teaching mode will restrict the efficiency of engineering operation. If intelligent and flexible path planning algorithm is adopted, each trajectory planning is unpredictable, and self-collision will inevitably occur.

The experiment of self-collision detection is based on the engineering application of the manipulator, randomly selecting the joint position and posture, and combining the working beat of the manipulator. When the joint angles of the manipulator are adjusted as shown in table 2, self-collision of the manipulator occurs.

The points $P0$ to $P7$ represent the endpoints coordinates of the central axis of the manipulator link, and points A to H represent the vertex coordinates of the bounding box of the G10 manipulator or load, which are the key points mentioned in section 3. Among them, Double-UR5 is divided into left arm (L-L) and right arm (R-L). Considering the average values, the time consumption for two single arms SR5A and SA10C is less than 12 ms, the self-test time of a 6-DOF manipulator is between 10 and 15 ms, and the test times for the two arms are 64.22 and 73.05 ms respectively. Thus, the double-UR5 test is equivalent to two 6-DOF self-collision tests and a cross test; and group 4 is equivalent to a one 6-DOF self-collision test and a cross detection. Due to the limitation of configuration design, the 4-DOF G10 will not undergo self-collision, but there is the possibility of collision between the link and load or the Zu3 manipulator. Therefore, this work establishes the corresponding rectangular bounding box for the G10 and contains some detection of edges. By comparing the results of collision experiments and practical engineering experience, the self-collision of a single arm occurs between the end-effort and the initial link, which accounts for a large proportion of incidents, as shown in the figures 4 and 5. It is mainly affected by the configuration design of the manipulator, and the algorithm adjusts the collision matrix to

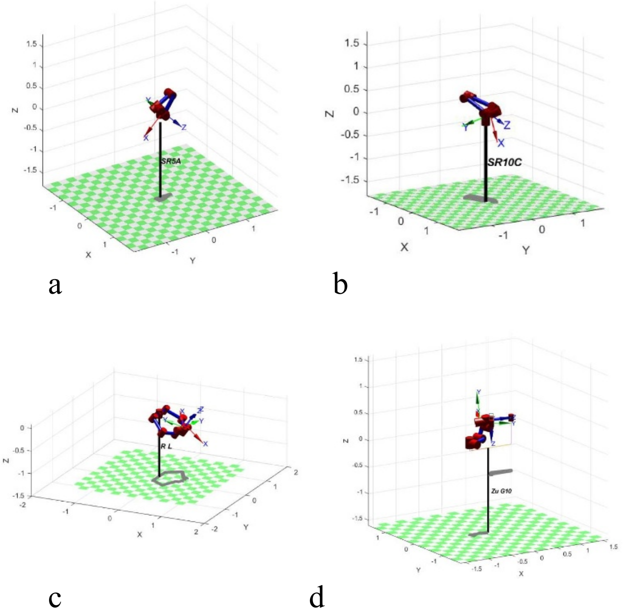


Figure 4. Self-collision detection pose: (a) SR5A, (b) SR10C, (c) Double-UR5, and (d) Zu3 & G10.

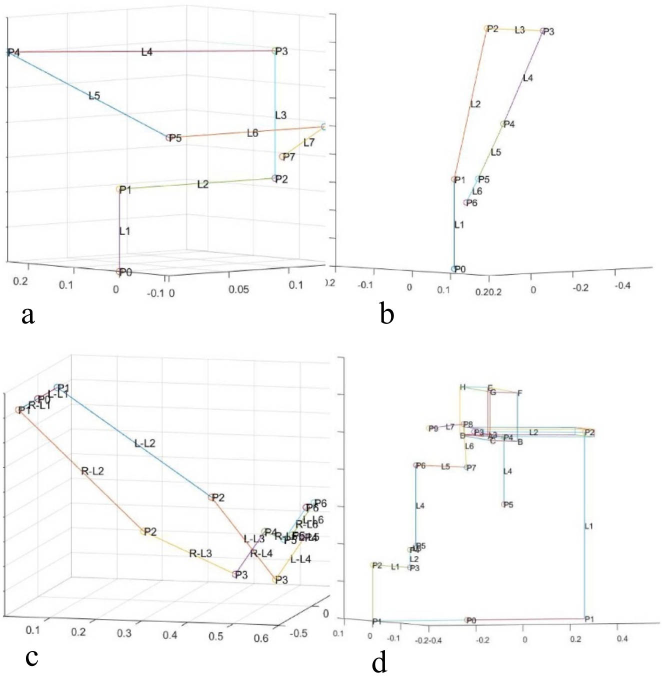


Figure 5. Self-collision detection structure: (a) SR5A, (b) SR10C, (c) Double-UR5, and (d) Zu3 & G10.

Table 3. Self-collision detection with load.

Group No.	Model	Joint angle (°)	Link	Key points	x	y	z	d (mm)	Time (ms)
1	SR5A	−35, −87, −147,	L5	P4	0.0189	−0.0132	0.6759	5.12	58.35
		123, −116, 92		P5	−0.2528	0.1770	0.9169		
			CD	C	−0.4713	−0.1495	0.4262		
				D	0.1097	−0.0112	1.0169		
2	SA10C	10.2, −134, 30,	L1	P1	0	0	0.1600	3.6	47.28
		10.2, 96, −21.8		P2	−0.3931	−0.0707	0.5736		
			GH	G	−0.1392	−0.2678	0.3681		
				H	0.0516	0.8882	0.6272		
3	Double-UR5	−57.4, 151, 28, 0,	R-L3	R-P2	0.3132	−0.0831	0.2003	5.78	65.23
		−64.8, 25.		R-P3	0.6434	−0.0871	0.4115		
		90, 25.2, 97.2,	Ball	L-P6′	0.5968	0.1225	0.1091		
		87.5, 0, 14.4			0.0215	0.0370	0.3787		
4	Zu3-G10	−18, −18.5, 21,	FG	F	0.0333	0.0112	0.2213	0.12	88.93
		66, 10.8, 70.		G	−0.0085	0.074	0.3200		
		36.7, −78, 0.018, 0	Ball	P5′	0.0189	−0.0132	0.6759		
					−0.2528	0.1770	0.9169		

reduce unnecessary computational steps. The probability of self-collision will be higher in the case of double arm cooperation, but the collision between each other is not limited to this.

4.3. Self-collision detection with load

Similar to the self-collision detection mentioned above, the self-collision detection with load is an important research aspect, and it is also a prominent problem to be solved in engineering application, especially for the operation of double-arms or cooperative manipulators.

The load may be a special end-effort tool that has a certain shape, or a target to be operated. As shown in table 3, when four groups of joint angles are so adjusted, a corresponding collision occurs.

The suction cup is a common end tool in the handling operation of engineering projects, and there is a certain distance between the load to be operated and the end coordinate system. In this study, the end offset of 0.02 is set for SR5A, the left arm of Double-UR5 and the load of Zu3, and the value can be adjusted with the model of the tool. In the four groups of load collision detection process, the time consumption is quite

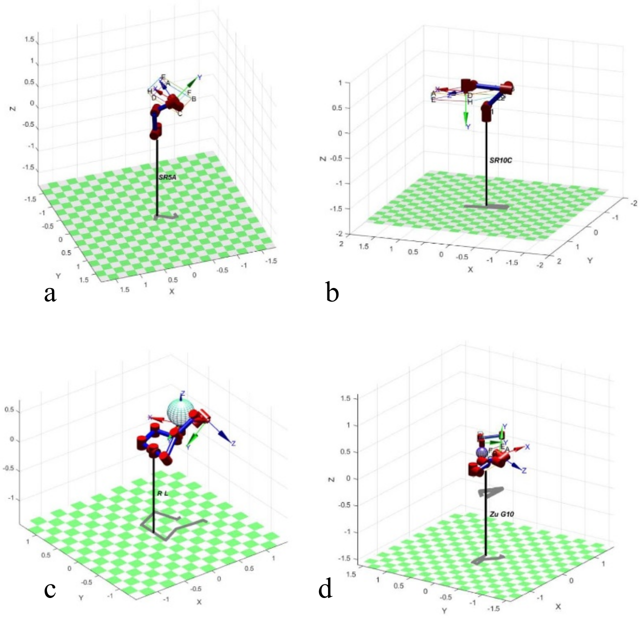


Figure 6. Self-collision detection pose with load: (a) SR5A, (b) SR10C, (c) double-UR5, and (d) Zu3 & G10.

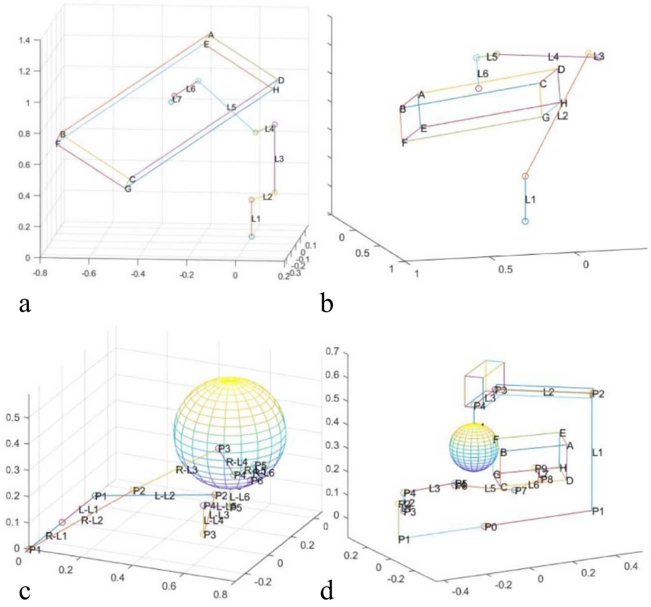


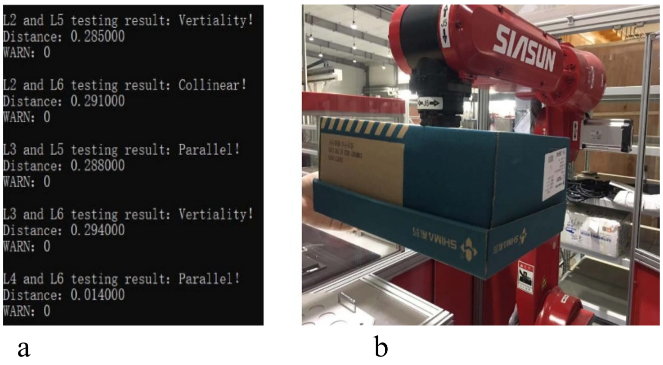
Figure 7. Self-collision detection structure with load: (a) SR5A, (b) SR10C, (c) double-UR5, and (d) Zu3 & G10.

different. The time consumption here includes not only the collision detection time with the load, but also the collision detection consumption of the manipulator's own joint link. Among them, SR10C is the first link L1 that collides with the load, and compared with L5 of SR5A, the collision occurs earlier. The average time consumption of the algorithm is about 12.6 ms. In fact, we should consider whether the z-axis coordinates of each key point of the manipulator and load are less than 0, that is, whether there is a collision between the manipulator and the working platform or the arm of the mobile manipulator collides with the vehicle body.

As shown in figures 6 and 7, the L-P6′ and P5′ in the third and fourth groups represent the offset coordinates of the end coordinates. The L3 of the right arm of Double-UR5 collides with the spherical load on the left arm, and the computation time in the table is 65.23 ms, mainly because the detection of spherical obstacles is easier than that of cuboids, and the algorithm is simple. The collision of the fourth group occurs between the loads at the end of the two cooperative manipulators, and the load shape is different.

Only the first group of SR5A of self-collision detection gives rise to coplanarity alarms. Generally speaking, manipulators with 90 degree joint bending such as SR5A, UR5 and Zu3 are more likely to be vertical, parallel, collinear, and coplanar in the application, as shown in figure 8(a). As shown in figure 8(b), from the collision test of the SR10C load in actual operation, each joint also participates in the work, the coordinate values will change, and noncoplanar positions account for more than 90% of the duration.

Therefore, in the detection process of the algorithm, these collision test is carried out first, which can also effectively reduce computation requirements.



```
L2 and L5 testing result: Vertiality!
Distance: 0.285000
WARN: 0

L2 and L6 testing result: Collinear!
Distance: 0.291000
WARN: 0

L3 and L5 testing result: Parallel!
Distance: 0.288000
WARN: 0

L3 and L6 testing result: Vertiality!
Distance: 0.294000
WARN: 0

L4 and L6 testing result: Parallel!
Distance: 0.014000
WARN: 0
```

Figure 8. Experimental testing: (a) algorithm test and (b) SR10C load carton test.

4.4. Comparison and analysis of algorithm

Collision detection has always been the focus of research in the field. Compared with the available literature, the algorithm proposed in this paper has improved characteristics as shown in table 4. The conventional configuration design can effectively avoid the collision between link and joint, but this will greatly reduce the reverse solution rate of the end of manipulator and limit the working range. The torque transformation and current detection methods are only effective post collision and do not give an early warning. Sensor methods will increase the cost affect operation accuracy. Self-CCD (continuous collision detection) is a fast collision detection library for deforming objects, but it is not suitable for the common industrial manipulator. DCD (discrete collision detection) realizes the detection requirements by updating the detection object and enabling both broad and narrow phase detection. For the average time consumption of 6-DOF self-collision detection,

Table 4. Comparison of the collision detection methods.

Detection method	Method list	Advantages	Disadvantages	Detection sequence
Configuration constraints	Rod design constraints, transformation threshold	Higher safety, self-collisions are almost non-existent	Lower solution rate, restricted working space	Before collision
Voltage or current variation	Perception skin, torque measurement, envelope lines	Responsive, more interactive applications	Complex wiring, poor anti-interference	After collision
Sensor	Lidar, visual, force, deformeter	Real-time planning, strong interaction	Increase cost, more calculations	Before/after collision
Widely used library	OMPL, FCL, self-CCD, DCD	Relatively complete functions, higher effectiveness and robustness	Algorithm complexity, implementation difficulties	Before/after collision
Proposed algorithm	Spatial geometric position judgment	Simple and effective, more self-detection objects	Lower accuracy with complex manipulator configuration	Before collision

the time of DCD is about 15.2 ms, which increased by about 20% compared with the proposed algorithm (12.6 ms). Moreover, it needs to store the details of the individual pairs that may collide in the intermediate process. Some combined algorithms based on the geometric bounding method, such as Hierarchical Bounding Box + Space Subdivision or OBB (Oriented Bounding Box) + Triangular Mesh, the accuracy is improved, but the time consumption is 30.1 and 48.2 ms respectively. The proposed method is simple and effective, the order of collision detection is obtained from experimental study and engineering experience, and the process takes relatively less time. Both OMPL (Open Motion Planning Library) and PLC (Flexible Collision Library) are widely used motion planning and collision detection libraries, and they include many mature and classic geometric and control-based planner algorithms, and realize the visual operation process through MoveIt and Rviz. However, these are generally complex and require various sensors, and the detection targets are mostly the mobile robot or manipulator and environmental obstacles. There is little research on the manipulator itself and load.

The proposed algorithm could be applied not only for self-collision detection in the manipulator itself, but could also include the manipulator and its load or cooperative manipulator. And it used for different types and DOF of manipulators effectively.

5. Conclusion

In this paper, the collision detection problem for a multi-DOF manipulator is examined in-depth in actual working conditions. The proposed algorithm is simple and can effectively solve the self-collision problem of single arm and double arm cooperatives. At the same time, it can also undertake collision detection with the operating load to avoid the damage to the manipulator and load. In addition, the application of this algorithm can overcome the constraints of joint angle in the hardware configuration design of the manipulator and thus increase the working range of the joint. It can also improve the reverse solution rate, break through the shackles of the traditional teaching mode, and can widely use offline programming and path planning algorithm, so as to further improve the applicability and intelligence of the multi-DOF manipulator.

In subsequent studies focus is likely to be on combining it with the path obstacle avoidance algorithm in offline programming to ensure the safety of detection, and further improve and enhance the intelligence and efficiency of path planning and obstacle avoidance processing of the manipulator.

Data availability statement

The data generated and/or analysed during the current study are not publicly available for legal/ethical reasons but are available from the corresponding author on reasonable request.

Funding information

This work was supported by the Major Science and Technology Innovation Project of Shandong Province (2019jzzy010128) and the Science and Technology Foundation of Liaoning Province (20180520022).

Conflict of interest

The authors have no conflicts of interest to declare.

ORCID iD

Lianhao Zhang https://orcid.org/0000-0002-0879-7718

References

[1] Himmelsbach U B, Wendt T M, Hangst N and Gawron P 2019 Single pixel time-of-flight sensors for object detection and self-detection in three-sectional single-arm robot manipulators *Proc. 2019 3rd IEEE Int. Conf. on Robotic Computing* (*Naples, Italy, 25–27 February 2019*) pp 250–3

[2] Birjandi S A B, Kuehnm J and Haddadin S 2020 Observer-extended direct method for collision monitoring in robot manipulators using proprioception and IMU sensing *Robot. Autom. Lett.* **5** 954–61

[3] Zhang T, Chen S, Xu G, Wang C and Tan C 2019 Joint angle measurement of manipulator and error compensation based on an IMU sensor *IET J. Eng.* **23** 9001–5

[4] Liang W, Feng Z, Wu Y, Gao J, Ren Q and Lee T H 2020 Robust force tracking impedance control of an ultrasonic motor-actuated end-effector in a soft environment *Proc. 2020 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems* (*Las Vegas, NV, USA*, *24–30 October 2019*) pp 7716–22

[5] Liang B, Liang W and Wu Y 2020 Parameterized particle filtering for tactile-based simultaneous pose and shape estimation *Robot. Autom. Lett.* **7** 1270–7

[6] Zhang T and Hong J 2019 Collision detection method for industrial robot based on envelope-like lines *Ind. Robot.* **46** 510–7

[7] Alberto D L, Elisa T, Gianluigi P, Stefano G and Ruggero C 2019 proprioceptive robot collision detection through Gaussian process regression *Proc. 2019 American Control Conf.* (*Philadelphia, PA*, *10–12 July 2019*) pp 19–24

[8] Kabanov A A and Tokarev D A 2019 Self-collision avoidance method for a dual-arm robot *Proc. 2019 3rd Int. Conf. on Control in Technical Systems* (*St. Petersburg, Russia*, *30 October 2019–1 November 2019*) pp 273–6

[9] Cruz-Ortiz D, Chairez I and Poznyak A 2020 Robust control for master-slave manipulator system avoiding obstacle collision under restricted working space *IET Control. Theory Appl.* **14** 1375–86

[10] Zhou X, Xu Z and Li S 2019 Collision-free compliance control for redundant manipulators: an optimization case *Front. Neurorobot.* **13** 50

[11] Sangiovanni B, Incremona G P, Piastra M and Ferrara A 2020 Self-configuring robot path planning with obstacle avoidance via deep reinforcement learning *IEEE Control Syst. Lett.* **5** 397–402

[12] Park K M, Kim J, Park J and Park F C 2021 Learning-based real-time detection of robot collisions without joint torque sensors *Robot. Autom. Lett.* **6** 103–10

[13] Oh J, Hyoin B and Oh J H 2017 Analytic inverse kinematics considering the joint constraints and self-collision for redundant 7DOF manipulator *Proc. 2017 1st IEEE Int. Conf. on Robotic Computing* (*Taichung, Taiwan*, *10–12 April 2017*) pp 123–8

[14] Sucan I, Chitta S and Pooley A 2019 Collision request (available at: http://docs.ros.org/en/indigo/api/moveit_core/html/structcollision__detection_1_1CollisionRequest.html) (Accessed June 2019)

[15] Sucan I, Chitta S and Pooley A 2019 Collision result (available at: docs.ros.org/en/indigo/api/moveit_core/html/structcollision__detection_1_1CollisionResult.html) (Accessed June 2019)

[16] Kramar V, Kramar O and Putin A 2020 The predicting collision with external obstacles of dual-arm multi-link robot *Proc. 2020 Int. Russian Automation Conf.* (*Sochi, Russia*, *6–12 September 2020*) pp 68–72

[17] Zhao L, Jiang Z, Sun Y, Zhao J and Liu H 2021 Collision-free kinematics for hyper-redundant manipulators in dynamic scenes using optimal velocity obstacles *Int. J. Adv. Robot. Syst.* **18** 1–17

[18] Jing W, Deng D, Wu Y and Shimada K 2020 Multi-UAV coverage path planning for the inspection of large and complex structures *Proc. 2020 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems* (*Las Vegas, NV, USA*, *24–30 October 2020*) pp 1480–6

[19] Zhao W, Li X, Chen X, Su X and Tang G 2020 Bi-criteria acceleration level obstacle avoidance of redundant manipulator *Front. Neurorobot.* **14** 54