Article

Real-Time Kinematics-Based Self-Collision Avoidance Algorithm for Dual-Arm Robots

Maolin Lei ^{1,2,3}, Ting Wang ^{1,2,*}, Chen Yao ^{1,2}, Huan Liu ⁴, Zhi Wang ^{1,2,5} and Yongsheng Deng ³

¹ State Key Laboratory of Robotics, Shenyang Institute of Automation, Chinese Academy of Sciences, Shenyang 110016, China; maolin_lei@outlook.com (M.L.); cyao@sia.cn (C.Y.); wangzhi1@sia.cn (Z.W.)

² Institutes for Robotics and Intelligent Manufacturing, Chinese Academy of Sciences, Shenyang 110169, China

³ Department of Mechanical Engineering and Automation, Northeastern University, Shenyang 110819, China; yshdeng@mail.neu.edu.cn

⁴ Interdepartmental Center for Advances in Robotic Surgery (ICAROS), University of Naples Federico II, 80125 Naples, Italy; hzauliuhuan@gmail.com

⁵ Department of Control Engineering, Shenyang University of Technology, Shenyang 110870, China

* Correspondence: wangting@sia.cn

Received: 29 July 2020; Accepted: 22 August 2020; Published: 26 August 2020

Abstract: Self-collisions of a dual-arm robot system can cause severe damage to the robot. To deal with this problem, this paper presents a real-time algorithm for preventing self-collisions in dual-arm systems. Our first contribution in this work is a novel collision model built using discrete spherical bounding volumes with different radii. In addition, we propose a sensitivity index to measure the distance between spheres with different radii in real time. Next, according to the minimal sensitivity index between different spheres, the repulsive velocity is produced at the centers of the spheres (control points), which the robot uses to generate new motion based on the robot kinematic model. The proposed algorithm offers the additional benefits of a decrease in the number of bounding spheres, and a simple collision model that can effectively decrease the computational cost of the process. To demonstrate the validity of the algorithm, we performed simulations and experiments by an upper-body humanoid robot. Although the repulsive velocity acted on the control points, the results indicate that the algorithm can effectively achieve self-collision avoidance by using a simple collision model.

Keywords: dual-arm manipulation; self-collision avoidance; motion planning

1. Introduction

Dual-arm robotic systems are widely used in various settings to expand the working range compared with single-arm robotic systems. As described in [1,2], dual-arm systems are widely used in space robots and disaster environments. To perform complicated tasks, each arm is provided with different targets and trajectories. One of the challenges in dual-arm cooperation control is to avoid self-collisions, which dramatically constrain the working range and affect the continuous motion of each arm. In this paper, we propose a self-collision avoidance method, where a dual-arm robot has the ability to detect and avoid self-collisions, to deal with such problems.

For self-collision avoidance methods, the collision model is critical. In collision models, a model of the robot is approximated, and relative position information is obtained between the robot links. In a collision model, the distance between the two arms is calculated in real time. A function related to the minimum distance is used to prevent robot collisions.

A collision model is always formed by various convex hulls. In previous studies, a collision detection model was built using a large number of polygons [3,4]. To tightly fit the different volumes

of the robot, researchers have developed algorithms to build each convex polyhedral model. Examples include the tight-fitting oriented bounding box trees (OBB) algorithm [5], and the axis-aligned bounding boxes (AABB) algorithm [6]. Many algorithms have been proposed to calculate the distance in convex polyhedral models such as the fast and robust polyhedral collision detection (V-Clip) algorithm [7]. As the features of polyhedral-based collision model are vertices, edges, and faces [8], the computational cost to determine if a pair of polyhedrons are disjointed and to compute the distance between them is high [7]. To reduce the computational cost between different collision models, researchers have developed sphere-based collision models to replace polygon-based collision models [9–11]. In [9,10], a sphere-based collision model was built using a sweep sphere model. For the sweep sphere line, the distance between two sweep sphere lines is calculated. The minimum distance may have a discontinuous gradient when the pair of closest points is on the two shapes [11]. In [11], an improved method was proposed, where a collision model was built using finite discretized spheres with the same radius. Compared with the sweep sphere model, the finite discretized sphere collision model algorithm [11] only needs to measure the distance between the center of each sphere (all possible distances between different spheres), which has a low computational cost. However, due to the difference in the links for each arm, the model of the origin robot may not be able to be approximated using spheres with the same radius.

Self-collision avoidance methods can be divided into two categories: (i) on-line (real-time) self-collision avoidance [3,6,12–15] and (ii) off-line planning of a collision-free trajectory [9,16,17] for the robot. Real-time self-collision avoidance methods always rely on the minimum distance between collision pairs. Earlier researchers proposed a collision model to check a collision pair in real time, where the robot abruptly stops when the collision model predicts a collision [4,6]. However, this behavior may influence the stability of the robot. To improve the results, researchers have proposed the skeleton algorithm [13,18–20], which establishes an artificial repulsive potential field to generate the virtual repulsive force on the robot links. Meanwhile, the skeleton algorithm has already been applied on the The German Aerospace Center (DLR) humanoid manipulator Justin [13]. Based on the dynamic model of the robot, the robot can generate a smooth trajectory to avoid self-collision. Although these algorithms can achieve self-collision avoidance, they do not consider the performance of the arm end-effector. Compared to these traditional methods, some researchers have proposed a unified framework for detecting collisions in real time by using a series of sensors capable of estimating the joint torque and acceleration [14]. However, using sensors always increases the cost of the robot due to the more expensive hardware.

In off-line methods, a collision-free trajectory is planned in advance. Based on the rapidly-exploring random tree (RRT) [21] algorithm, researchers have proposed an algorithm to plan a collision-free path [22] for each arm in advance. The extended stochastic trajectory optimization for motion planning (STOMP) algorithm, which is an improvement of the STOMP [23] algorithm, obtains a qualitatively collision-free trajectory according to user preferences [24]. Some researchers have presented a method for generating and synchronizing path-constrained trajectories for multi-robot systems [25]. Regarding the Asea Brown Boveri (ABB) company, they proposed a method to monitor the multi-arm, whether local at the collision space at the same time [26], before planning the path of each arm. With the development of machine learning, researchers can set different collision regions using efficient data-driven techniques to predict the joint collision position based on a support vector machine (SVM) algorithm [16,17], which may lead to higher computational costs. In particular, the method proposed in [16] has already been applied on the Istituto Italiano di Tecnologia (IIT) SCAFoI humanoid robot. Compared with on-line methods, off-line planning can be dangerous. For example, a robot can be misguided in planning a collision trajectory. In off-line planning, the collision-free trajectory must be planned before the arm moves to the target, so off-line planning also leads to higher computational costs.

In this paper, we propose an optimized collision model that decreases the computational cost as well as the robot hardware cost. In addition, it increases the reaction speed and accuracy simultaneously without any sensors to detect the relative distance of the two arms. To demonstrate the validity of the

algorithm, two independent arms with conflicting tasks are considered, so that each arm is in danger of self-collision. The goal of this study is to develop a real-time self-collision avoidance method that consists of: (1) a collision model based on a finite number of spheres with different radii; and (2) a suitable self-collision avoidance strategy. The main contributions of this study are summarized as follows:

- (1) In this study, we developed a collision model by using multiple discretized spherical bounding volumes with different radii, which has fewer spheres and can enclose the robot with a low approximation error.
- (2) To calculate all possible distances between spheres with different radii, a sensitivity index was proposed to measure the distance between different spheres. According to the minimal sensitivity index, the magnitude of the repulsive velocity can be measured using the sigmoid function.
- (3) The repulsive velocity will generate on each control point of each link. Based on the kinematic model of each arm, the repulsive velocity on each control point is projected in the velocity direction of the end-effector, which helps the robot modify the magnitude of the end-effector velocity, while generating the repulsive velocity on the control points to avoid self-collision at the collision spheres.

The remainder of this paper has the following structure. In Section 2, the collision model of the dual-arm robot is developed, along with the method for calculating the minimum distance between the sphere bounding volumes. Section 3 describes the development of the real-time self-collision avoidance strategy of the robot. The repulsive velocity acts on the center of the spherical volumes to generate a smooth trajectory to avoid self-collision. In Section 4, the modeling parameters are identified in an example in which a robot is used. Simulations and experiments demonstrate the validity of the proposed algorithm. Section 5 concludes this work.

2. Collision Model of the Robot

2.1. The Dual-Arm Robot

The algorithm was implemented on the upper body of a humanoid dual-arm robot as illustrated in Figure 1a. The robot was designed by our group (the explosives ordnance disposal robot group of Shenyang Institute of Automation). The robot was equipped with two arms with seven degrees of freedom (7-DOF) and the torso with 2-DOF. Meanwhile, each arm was equipped with a gripper with 1-DOF as the end-effector.

Figure 1. (a) Humanoid dual-arm robot. (b) Structure of the robot with 16 degrees of freedom. The robot was divided into three parts: torso, left arm, and right arm.

To implement our algorithm, we first divided the robot into three parts: the left arm, the right arm, and the torso. Figure 1b shows the structure of the robot, where the dotted lines represent the joints of the robot; q_{li} , q_{ri} , and q_{ti} are defined as the i -th joint angle of the left-arm, the right-arm, and the torso,

respectively. To avoid collision of the links of the left arm, the angle limits of the i -th joint of the left arm were set as $[q_{li,l}, q_{li,u}]$. Similarly, the angle limits of the i -th joint of the right arm and torso were set as $[q_{ri,l}, q_{ri,u}]$ and $[q_{ti,l}, q_{ti,u}]$, respectively.

The frames are defined as follows. The base frame $\{S\}$ is defined as the world frame in which the robot is located. Frame $\{S_{ki}\}$ is defined as the i -th link frame of the robot part k ($k = t, l, r$, where t indicates the torso part, l the left-arm, and r the right-arm). The positions of the control points (center of each spherical volume) are defined according to these link frames. In addition, the position vector ${}^{ki}P \in R^3$ and the rotation matrices ${}^S R_{ki}$ are defined from $\{S\}$ to $\{S_{ki}\}$; J_{kp} are Jacobian matrices for control points p with respect to the $\{S_{ki}\}$ frame, as illustrated in Figure 2. In this way, a point expressed as $P_0 = (x_0, y_0, z_0)^T$ in the $\{S_{ki}\}$ frame can be expressed in the base frame as ${}^S P_0 = {}^S R_{ki} {}^{ki} P_0 + {}^S P_{kiORG}$. Thus, all the control points are expressed in the base frame $\{S\}$ to detect a collision in real time.

Point a is expressed in frame B as ${}^B P_a = (x_a, y_a, z_a)^T$ and in frame A as ${}^A P_a = (x'_a, y'_a, z'_a)^T$, whereas point b is expressed in frame C as ${}^C P_b = (x_b, y_b, z_b)^T$ and in frame A as ${}^A P_b = (x'_a, y'_a, z'_a)^T$, as illustrated in Figure 2b. These can be calculated as follows:

$${}^A P_a = {}^A R_B {}^B P_a + {}^A P_{BORG}, \quad (1)$$

$${}^A P_b = {}^A R_C {}^C P_b + {}^A P_{CORG}, \quad (2)$$

The distance between different points in each part can be computed as

$$d_{ab} = \sqrt{(x'_a - x'_b)^2 + (y'_a - y'_b)^2 + (z'_a - z'_b)^2} \quad (3)$$

Figure 2. (a) Frame for the dual-arm robot. (b) Calculation of the distance d_{ad} between point a and point b .

2.2. Collision Model

The main propose of the collision model is to evaluate the minimum distance between two arms. After the model is built, the minimum distance between elements in the collision model must be evaluated. The function related to the minimum distance prevents the robot from colliding.

Based on the collision model using discretized spheres with the same radius [21], the method used in our robot arms is illustrated in Figure 3c. In this work, we proposed building a collision model using discretized finite spherical volumes with different radii at specific positions (control points) as illustrated in Figure 3d. Comparing Figure 3c with Figure 3d, the later collision model can be closer to the original arm. In this way, the collision model of each link of each arm can be described using a few different spheres. In general, we always used the same radius spheres at the same link and chose the

size of each sphere according to the size of each link, which is illustrated in Figure 3a. The detection of all possible distances between different control points with different spheres is illustrated in Figure 3b.

Figure 3. Collision model of the right arm, showing the distance between sphere bounding volumes. (a) the robot arm; (b) the distance between different spheres with different radii; (c) building a collision model using discretized finite spherical volumes with same radii at specific positions; (d) building a collision model using discretized finite spherical volumes with different radii at specific positions.

The minimum distance between these bounding volumes produces the repulsive velocity. As illustrated in Figure 3, distance d_{ab} represents the distance between different spheres with different radii. To detect all the possible distances between different spheres and to calculate the minimum distance with different pairs of spheres, we propose the sensitivity index β ($\beta > 0$) and define it in Equation (4) to measure the collision between different spheres according to two spherical volumes with different radii r_a and r_b ($r_a < r_b$).

$$\beta_{ab} = \frac{d_{ab}}{r_a + r_b}, \quad (4)$$

$$\begin{cases} \beta_{ab} < 1, \text{ collision} \\ \beta_{ab} = 1, \text{ at the moment of collision} \\ \beta_{ab} > 1, \text{ no collision} \end{cases}, \quad (5)$$

The minimum distance between these bounding volumes produces the repulsive velocity. As illustrated in Figure 3, distance d_{ab} represents the distance between different spheres with different radii. To detect all the possible distances between different spheres and to calculate the minimum distance with different pairs of spheres, we proposed the sensitivity index β ($\beta > 0$) and defined it in Equations (4) and (5) to measure the collision between different spheres according to two spherical volumes with different radii r_a and r_b ($r_a < r_b$), as illustrated in Figure 4.

$$\beta_{lr} = \begin{bmatrix} \beta_{11} & \beta_{12} & \dots & \beta_{1m} \\ \vdots & \vdots & \ddots & \vdots \\ \beta_{n1} & \beta_{n2} & \dots & \beta_{nm} \end{bmatrix}, \quad (6)$$

In Equations (7)–(9), β_{lrmin} denotes the minimum element in the matrix:

$$\beta_{lrmin} = \min(\beta_{lr}), \quad (7)$$

Figure 4. The significance of the sensitivity index β .

3. Self-Collision Avoidance Strategy

Our self-collision avoidance strategy relies on relationships between the minimal sensitivity index, the repulsive velocity, and the angular velocity of the arm. Section 3.1 illustrates the relationship between the minimal sensitivity index and the repulsive velocity, which shows how to use the repulsive velocity according to the minimal sensitivity index. Section 3.2 illustrates the relationship between the repulsive velocity and the velocity at the end-effector of each arm. Section 3.3 shows that to solve self-collision using our algorithm, each arm must be controlled at the joint velocity level.

3.1. Repulsive Velocity between Two Spheres

Each arm of the robot has multiple bounding volumes, and the minimal sensitivity index β_{lrmin} indicates the relationship between the closest bounding volumes. When $\beta_{lrmin} \leq 1$, the repulsive velocity v_{re} is produced at these control points.

Meanwhile, inspired by the on-line method of human–robot collision avoidance algorithms in [27], we chose the sigmoid function to measure the magnitude of the repulsive velocity. By using the sigmoid function, the magnitude of the repulsive velocity $\|v_{re}\|$ can be calculated as

$$\|v_{re}\| = \frac{V_{max}}{1 + e^{(\alpha \times (\beta_{lrmin} - 0.7))}}, \quad (8)$$

where V_{max} is the maximum velocity allowed at the end of the robot and α is the shape factor, as described in [27]. If $V_{max} = 0.25$ and $\alpha = 20$, the profile shown in Figure 5a is achieved, which illustrates the relationship between the minimal sensitivity index and the repulsive velocity.

Figure 5. (a) Relationship between minimal sensitivity index and repulsive velocity. (b) The repulsive velocity produced at the center of the spherical volumes.

From the profile shown in Figure 5, the sigmoid function not only ensures the continuity of the speed change, but when the relative distance of the arm becomes smaller, the gradient of the repulsive change continues to be changed. In particular, in Figure 5, a shorter β_{lrmin} resulted in a larger repulsive velocity, helping the robot avoid self-collision. Meanwhile, we can conclude that when $\beta_{lrmin} = 0.4$, the magnitude of the repulsive velocity $\|v_{re}\| = 0.25$ is equal to the maximum velocity V_{max} of the end-effector. This can also define the radii of each sphere.

Considering each link of the robot as the cylinder and the radius of each cylinder (link) is $r_{cy,li}$, the radius r_{cm} of each sphere can be defined according to $\frac{r_{cy,li}}{r_{cm}} = 0.4$. In this way, we can define the radius of each sphere. When $\beta_{lrmin} = 0.4$, it indicates a collision between the arms; when the magnitude of the repulsive velocity $\|v_{re}\| = V_{max} = 2.5$, it indicates that the robot will stop motion.

As illustrated in Figure 5b, the position of the control point a , on the left, is defined as $(x'_a, y'_a, z'_a)^T$, which is expressed in the base frame, and the other point b , on the right, is $(x'_b, y'_b, z'_b)^T$. Then, the repulsive velocity direction can be defined as

$$\begin{cases} \vec{ab} = \frac{x'_a y'_a z'_a - x'_b y'_b z'_b}{d_{ab}} \text{ direction on point } a \\ \vec{ba} = \frac{x'_a y'_a z'_a - x'_b y'_b z'_b}{d_{ab}}, \text{ direction on point } b \end{cases} \quad (9)$$

The repulsive velocity produced at the control points can be computed as:

$$\begin{cases} v_{re,a} = \vec{ab} \|v_{re}\|, \text{ repulsive velocity on point } a \\ v_{re,b} = \vec{ba} \|v_{re}\|, \text{ repulsive velocity on point } b \end{cases} \quad (10)$$

Section 3.1 introduced the calculation of the repulsive velocity between two spheres. In reality, the strategy must be applied on the dual-arm system in 3D space.

Our approach is illustrated on the dual-arm robot in Figure 6, which has a left arm (l) and a right arm (r). Each arm has a different original velocity on the end-effector, which is defined as $v_{end,k}$, where $k = l$ or r represents the left or right arm. The original joint velocity is defined as $\dot{\theta}_{ki}$, where i represents the i -th joint of each arm. Thus, the joint velocities $\dot{\theta}_k = [\dot{\theta}_{k1} \dots \dot{\theta}_{kj-1} \dot{\theta}_{kj}] \in R^j$ can be calculated as follows:

$$\dot{\theta}_k = J_k^\# \begin{bmatrix} v_k \\ \omega_k \end{bmatrix}, \quad k = l \text{ or } r, \quad (11)$$

where $J_k = [J_v \ J_\omega]^T$ are the Jacobian matrices with respect to the end-effector of the robotic arms, and $J_k^\#$ is the pseudo-inverse; $v_k \in R^3$ and $\omega_k \in R^3$ represent the linear and angular velocity of the end-effector of each arm, respectively.

Figure 6. Two redundant arms work in 3D space.

3.2. Repulsive Velocity at the End-Effector

As illustrated in Figure 6, a bounding sphere is centered at point of the left arm and at point b of the right arm of the robot. These control points both have repulsive velocity $v_{rv,k}$ ($k = l$ or r) when $\beta_{ab} = \beta_{lrmin} = \min(\beta_{lr}) \leq 1$, which implies that these spherical bounding volumes will collide. The repulsive velocity acts on the centers of the spheres, as discussed in Section 2.

The repulsive velocity needs to be controlled at the joint velocity level. To determine the repulsive velocity of the control point, only the joints prior to this control point should be given a repulsive joint velocity. In the case shown in Figure 6, the repulsive joint velocity is generated on joints 1–5 of the left arm, and on joints 1–7 of the right arm.

Defining part of the repulsive joint velocity of the k arm from joint 1 to joint i ($i \leq j$) as $\dot{q}_{rep,k} = [\dot{q}_{1k} \dots \dot{q}_{ik}]^T \in R^i$ ($k = l$ or r) of the arm, the repulsive joint velocity of this part of the arm can be calculated as follows:

$$\dot{q}_{rep,k} = J_{rv,k}^{\#} \begin{bmatrix} v_{rv,k} \\ \omega_{rv,k} \end{bmatrix}, \quad (k = l \text{ or } r), \quad (12)$$

where $J_{re,k} = [J_{re,v} \ J_{re,\omega}]^T \in R^{6 \times i}$ are Jacobian matrices for the control points, as illustrated in Figure 2a in arm k ($k = l$ or r). Here, only the positions of the two control points are controlled, so we set $\omega_{re,k} = [0 \ 0 \ 0]^T \in R^3$ in arm k ($k = l$ or r).

The velocity of the joints after/distal to the control point was set to zero. In this way, the repulsive joint velocity $\dot{q}_{re,k}$ ($k = l$ or r) can be obtained as

$$\dot{q}_{re,k} = [\dot{q}_{repar,k} \ 0 \dots 0] \in R^j, \quad (13)$$

According to the repulsive joint velocity, the end-effector velocity of the arm is generated with an end-effector repulsive velocity $v_{re,k,end}$ as

$$v_{re,k,end} = J_k \dot{q}_{re,k}, \quad (k = l \text{ or } r), \quad (14)$$

Due to the collision model, which was built using discretized spheres with different radii and detecting the minimum distance between the spheres, the minimum distance points could change instantaneously, causing discontinuous velocities and non-smoothness in the overall behavior of the system. In this way, we chose to project the velocity in the direction of the end-effector velocity, which only changes the magnitude of the end-effector velocity, to alleviate this situation. Meanwhile, this strategy can also generate the repulsive velocity.

3.3. Real-Time Self-Collision Avoidance

As illustrated above, the end-effector repulsive velocity $v_{re,k,end}$ calculated by (14) can have different directions from the original end-effector velocity $v_{end,k}$ ($k = l$ or r), which is planned in advance for a dual-arm task. The end-effector repulsive velocity can drive the effector to an uncertain direction, and the desired task may not be finished. To solve this problem, the algorithm modifies only the magnitude of the original end-effector velocity and keeps the direction unchanged.

Using the end repulsive velocity $v_{re,k,end}$ calculated in (14), the end-effector repulsive velocity $v_{re,k,end}$ ($k = l$ or r) is then projected to the original velocity of the end-effector $v_{end,k}$ ($k = l$ or r), which is represented by the red lines in Figure 6.

The angle γ_k between $v_{end,k}$ and $v_{re,k,end}$ ($k = l$ or r) and can be calculated as

$$\gamma_k = \arccos \left(\left(\frac{v_{re,k,end}}{\|v_{re,k,end}\|} \right)^T \frac{v_{end,k}}{\|v_{end,k}\|} \right), \quad (k = l \text{ or } r), \quad (15)$$

The projected repulsive velocity $v_{pro,k}$ ($k = l$ or r) can be calculated as

$$v_{pro,k} = \|v_{re,k,end}\| \cos \gamma_k \frac{v_{end,k}}{\|v_{end,k}\|}, \quad (k = l \text{ or } r), \quad (16)$$

Summing the projected repulsive velocity $v_{pro,k}$ and the original end-effector velocity $v_{end,k}$, the new velocity of the end-effector $v_{new,k}$ ($k = l$ or r) is obtained as

$$v_{new,k} = v_{end,k} + v_{pro,k}, \quad (k = l \text{ or } r), \quad (17)$$

Next, the new joint velocity can be calculated using (18), where $\dot{\theta}_{new,k} = [\dot{\theta}_{k,1} \dots \dot{\theta}_{k,j-1}, \dot{\theta}_{k,j}] \in R^j$ ($k = l$ or r) are the new joint velocities:

$$\dot{\theta}_{new,k} = J_{rv,k}^{\#} \begin{bmatrix} v_{new,k} \\ 0 \end{bmatrix}, \quad (k = l \text{ or } r), \quad (18)$$

4. Simulation and Application of the Algorithm

This section presents the simulation and experiment for self-collision avoidance by a robot using the proposed algorithm. The robot has dual 7-DOF arms, as shown in Figure 1. To avoid collision of the links with one arm, the joints were set as indicated in Table 1. According to the specifications of the robot, the maximum velocity of the end-effector was below 2.5 m/s.

The collision model was constructed with 10 spheres centered at 10 control points on each arm, as illustrated in Figure 7. The red lines in Figure 7 represent the distances between the control points on different arms. The diameters of the spheres are listed in Table 2.

As the collision model is built, the sensitivity index β_{lr} is calculated according to Section 2. The new joint velocities $\dot{\theta}_{new,k} = [\dot{\theta}_{k,1} \dots \dot{\theta}_{k,j-1}, \dot{\theta}_{k,j}] \in R^j$ ($k = l$ or r) are then calculated for self-collision avoidance when one spherical volume collides with another spherical volume.

Table 1. Constraints of joints.

Joint	1	2	3	4	5	6	7
Joint limit (degrees)	−180, 25	−270, 90	−270, 90	−220, 45	−180, 180	0, 100	−90, 90

Table 2. Sphere diameters.

Point	1	2, 3	4	5, 6	7	8	9, 10
Link	1	2	3	4	5	6	7 (end-effector)

Figure 7. Collision model of the robot used to calculate the distances of the collision model. The red lines represent the distance from the two points on the right arm to all points on the left arm.

4.1. Simulations

The simulation was performed using an Intel Atom® E3845 processor (Quad Core, 1.91 GHz, 10 W) with 32 GB of RAM, and Ubuntu 16.04 using ROS Kinetic. The simulation results are shown in Figure 8.

The algorithm was simulated on a dual-arm robot with 10 control points on each arm. First, the system needs to detect and update the matrix in real time. Following the steps presented in Section 3, the new joint velocities were calculated, which changes the motion of the robot to avoid collision. These processes are illustrated in Figure 8.

Figure 8. Cont.

Figure 8. Process of avoiding self-collision in the dual-arm system. **(a.1)** Left arm collides with right arm; the repulsive velocity acts on the end of the arm. **(a.2)** Left arm collides with right arm; the repulsive velocity acts on link 5 on the right arm and on the end-effector on the right arm. **(b.1)** Left arm collides with right arm; the repulsive velocity acts on link 6 on the left arm and on the end-effector on the right arm. **(b.2)** Left arm collides with right arm; the repulsive velocity acts on link 6 on the right arm and on the end-effector on the left arm. **(b.3)** Left arm collides with right arm; the repulsive velocity acts on link 6 on the right arm and on link 6 on the left arm. The repulsive vector ensures that the trajectory of the robot remains collision free.

In Figure 8(a.i, $i = 1, 2$), we intentionally set the same target positions for the two arms (i.e., without the self-collision avoidance algorithm, the two arms will collide as the end-effectors approach their targets). Actually, with the self-collision avoidance strategy, the repulsive velocities are calculated on the control points. In Figure 8(a.1), the repulsive velocity acted on the end-effector of the left arm and link 6 of the right arm; in Figure 8(a.2), the repulsive velocity acted on the end-effectors of the robotic arms. Both the left and right arms changed their motion to avoid collision. The repulsive velocity on each arm both act on the end-effectors, which makes each arm move away from their target.

In Figure 8(b.i, $i = 1, 2, 3$), we intentionally set the left and right end-effectors with different targets. Without the self-collision avoidance algorithm, the two arms will collide as the end-effectors approach their targets. With the self-collision avoidance strategy, the repulsive velocities were calculated on the control points. In Figure 8(b.1), the repulsive velocity acted on link 7 of the left arm and link 6 of the right arm; in Figure 8(b.2) the repulsive velocity acted on the end-effector of the left arm and link 6 of the right arm; in Figure 8(b.3) the repulsive velocity acted on link 6 of the left arm and link 6 of the right arm. The simulated process of avoiding self-collision is illustrated in Figure 8(b.i, $i = 1, 2, 3$). The right arm and left arm both changed their trajectories to avoid self-collision, but one of the dual-arms was also close to the target in the other trajectory. In Figure 8(b.1), the left arm was close to the target. In Figure 8(b.2), the left arm was close to the target. In Figure 8(b.3), the right arm was close to the target.

A common approach for self-collision avoidance is the artificial potential field in which one of the arms is modeled with an obstacle. However, these strategies often result in a (topologically unnecessary) minimum [28]. Meanwhile, we completed two simulations using the traditional artificial potential field method. These methods do not consider the velocity vector of the end-effector. In Figure 9a, two arms of dual-arm had a common target and in Figure 10a, two arms of dual-arm had different targets. The trajectories of each arm illustrated that the two arms both easily fell into the local minimum and vibrated around the local minimum points, as illustrated in Figure 9b, Figure 10b, respectively.

(a) Initial state (left) and final state (right)

(b) Trajectory of the end-effector of each arm

Figure 9. Process of avoiding self-collision in the dual-arm system using traditional artificial potential field method. Two arms of the dual-arm had a common target. (a) The initial state and final state of the robot; (b) Trajectory of the end-effector of each arm.

Compared with the traditional artificial potential field method, the method of this paper improved the problems of the traditional method due as we considered the action of the end-effector. Meanwhile, the simulation results of the different tasks described above prove the effectiveness of this real-time self-collision algorithm based on the kinematic model of a robot. The algorithm uses a simple collision model and the distance calculation method to decrease the computational costs of the algorithm.

(a) Initial state (left) and final state (right)

(b) Trajectory of the end-effector of each arm

Figure 10. Process of avoiding self-collision in the dual-arm system using the traditional artificial potential field method. Two arms of the dual-arm had different targets. (a) The initial state and final state of the robot; (b) Trajectory of the end-effector of each arm

4.2. Experimental Results

The algorithm was implemented on a dual-arm robot in an office environment, and demonstrated the change of the end-effector velocity. The sampling frequency of the control loop was set at 50 Hz.

Figures 9 and 10 show the results of the two tasks. In Figure 9, the two arms of the dual-arm robot had a common target, while in Figure 10, each arm had a different target.

Figure 11a shows the snapshots of the experiment. The velocity profile and the trajectory of the end-effect of the arms are shown in Figure 11b,c, respectively. As the two arms had the same target, both arms changed trajectory to avoid self-collision. In fact, both arms moved away from the target.

In the second experiment, the robotic arms had different targets, as illustrated in Figure 12a. The velocity profile of the end-effector and the trajectories of the two arms are shown in Figure 12b,c, respectively.

(a) Two arms of the dual-arm robot have a common target and the experiments are illustrated in two views

(b) Velocity profile of the end-effector of each arm

Figure 11. Cont.

(c) Trajectory of the end-effector of each arm

Figure 11. Experiment to assess the ability of a dual-arm system to avoid self-collision when the left and right arms have the same target. (a) The first experiment; (b) Velocity profile of the end-effector of each arm; (c) Trajectory of the end-effector of each arm.

(a) Two arms of the dual-arm robot have different targets; the experiments are illustrated in two views.

(b) Velocity profile of the end-effector of each arm

Figure 12. Cont.

(c) Trajectory of the end-effector of each arm

Figure 12. Experiment to assess the ability of the dual-arm system to avoid self-collision when the left and right arms had different targets. (a) Two arms of the dual-arm robot have different targets; The experiments are illustrated in two views; (b) Velocity profile of the end-effector of each arm; (c) Trajectory of the end-effector of each arm.

As the two arms had different targets, each arm changed its motion to find a collision-free trajectory. As illustrated in Figure 12, the repulsive velocity produced the end-effect on the left arm and acted on link 6 of the right arm. In this way, the repulsive velocity was projected onto the end-effect of the left arm. In other words, both arms in the dual-arm system changed their motion to avoid self-collision: the right arm changed its motion to approach the target, whereas the left arm changed its motion by moving away from the target and the right arm.

Meanwhile, as illustrated in Figure 11b, Figure 12b, respectively, although there were fluctuations in the end-effector velocity, the fluctuation did not affect the results of the experiment.

These experiments were similar to the simulations, which are described in Section 4.1, and demonstrated the effectiveness of our algorithm in simulation A (robotic arms with the same target) and simulation B (robotic arms with different targets). Meanwhile, these demo of the experiments were upload on the web as illustrate in Supplementary Materials).

5. Conclusions

This paper proposed an algorithm based on the kinematic model of a robot to enable self-collision avoidance in real time. The collision model was based on a finite number of discrete spherical bounding volumes with different radii. In addition, the algorithm proposed a sensitivity index for measuring the collisions between spheres. When the bounding spheres collide, two repulsive velocities are generated on the control points, and the motion of the arms are changed accordingly to avoid collisions between the two arms. Simulations and experiments were conducted on a dual-arm robot to verify the effectiveness of the algorithm. The results indicate that the algorithm can effectively avoid self-collision between arms using a simple collision model. The method optimizes the collision model, decreases the computational cost, and increases the reaction speed as well as accuracy. This algorithm can be used in tele-operation and fully autonomous control systems.

In the future, we aim to devise a strategy/framework that can avoid self-collision and accomplish the desired tasks in concert, similar to human motion. Meanwhile, we aim to develop this strategy to avoid obstacles when external environment changes are detected to deal with more complicated

tasks. In addition, a strategy for self-collision avoidance of multi-arm (>2) robots is also a direction worth investigating.

Supplementary Materials: The following are available online at <http://www.mdpi.com/2076-3417/10/17/5893/s1>, Video S1: Real-Time Kinematics-Based Self-Collision Avoidance Algorithm for Dual-Arm Robots.

Author Contributions: Conceptualization, M.L. and T.W. methodology, M.L.; software, M.L.; validation, M.L. and Z.W.; formal analysis, M.L. and T.W.; investigation, M.L. and T.W.; resources, T.W., C.Y. and Y.D.; data curation, M.L. and T.W.; writing—original draft preparation, M.L.; writing—review and editing, M.L., T.W. and H.L.; visualization, M.L. and H.L.; supervision, T.W., C.Y. and Y.D.; project administration, T.W., C.Y.; funding acquisition, T.W., C.Y. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by National Key R&D Program of China (Grant Nos. 2016YFE0206200) and LiaoNing Revitalization Talents Program (XLYC1807018).

Acknowledgments: The authors would like to thank Shiliang Shao and Yun Su for their help in developing the control system of the upper-body humanoid dual-arm robot, and would like to thank Yao Xu, Yongliang Wang, Hengliang Xie, and Yingbiao Wu for their support in designing and testing the robot.

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

References

1. Zhang, X.; Liu, J.; Feng, J.; Liu, Y.; Ju, Z. Effective Capture of Nongrasable Objects for Space Robots Using Geometric Cage Pairs. *IEEE/ASME Trans. Mechatronics* **2020**, *25*, 95–107. [\[CrossRef\]](#)
2. Klamt, T.; Kamedula, M.; Karaoguz, H.; Kashiri, N.; Laurenzi, A.; Lenz, C.; Leonardis, D.; Hoffman, E.M.; Muratore, L.; Pavlichenko, D.; et al. Flexible Disaster Response of Tomorrow: Final Presentation and Evaluation of the CENTAURO System. *IEEE Robot. Autom. Mag.* **2019**, *26*, 59–72. [\[CrossRef\]](#)
3. Okada, K.; Inaba, M.; Inoue, H. Real-time and Precise Self Collision Detection System for Humanoid Robots. In Proceedings of the 2005 IEEE International Conference on Robotics and Automation, Barcelona, Spain, 18–22 April 2005; pp. 1060–1065. [\[CrossRef\]](#)
4. Okada, K.; Inaba, M. A Hybrid Approach to Practical Self Collision Detection System of Humanoid Robot. In Proceedings of the 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems, Beijing, China, 9–15 October 2006; pp. 3952–3957.
5. Gottschalk, S.; Lin, M.C.; Manocha, D. OBBTree: A hierarchical structure for rapid interference detection. In Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques, New Orleans, LA, USA, 4–9 August 1996; pp. 171–180.
6. Xiao-Rong, W.; Meng, W.; Chun-Gui, L. Research on Collision Detection Algorithm Based on AABB. In Proceedings of the 2009 Fifth International Conference on Natural Computation, Tianjian, China, 14–16 August 2009; Volume 6, pp. 422–424.
7. Mirtich, B. V-Clip: Fast and robust polyhedral collision detection. *ACM Trans. Graph.* **1998**, *17*, 177–208. [\[CrossRef\]](#)
8. Chang, J.-W.; Wang, W.; Kim, M.-S. Efficient collision detection using a dual OBB-sphere bounding volume hierarchy. *Comput. Des.* **2010**, *42*, 50–57. [\[CrossRef\]](#)
9. Kakiuchi, Y.; Ueda, R.; Kobayashi, K.; Okada, K.; Inaba, M. Working with movable obstacles using on-line environment perception reconstruction using active sensing and color range sensor. In Proceedings of the 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems, Taipei, Taiwan, 18–22 October 2010; pp. 1696–1701.
10. Cascio, J.; Karpenko, M.; Gong, Q.; Sekhavat, P.; Ross, I.M. Smooth proximity computation for collision-free optimal control of multiple robotic manipulators. In Proceedings of the 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems, St. Louis, MO, USA, 11–15 October 2009; pp. 2452–2457.
11. Selvaggio, M.; Abi-Farraj, F.; Pacchierotti, C.; Giordano, P.R.; Siciliano, B. Haptic-Based Shared-Control Methods for a Dual-Arm System. *IEEE Robot. Autom. Lett.* **2018**, *3*, 4249–4256. [\[CrossRef\]](#)
12. Dietrich, A.; Wimbock, T.; Albu-Schaffer, A.; Hirzinger, G. Integration of Reactive, Torque-Based Self-Collision Avoidance Into a Task Hierarchy. *IEEE Trans. Robot.* **2012**, *28*, 1278–1293. [\[CrossRef\]](#)
13. De Santis, A.; Albu-Schaffer, A.; Ott, C.; Siciliano, B.; Hirzinger, G. The skeleton algorithm for self-collision avoidance of a humanoid manipulator. In Proceedings of the 2007 IEEE/ASME International Conference on Advanced Intelligent Mechatronics, Zurich, Switzerland, 4–7 September 2007; pp. 1–6.

14. Vorndamme, J.; Schappler, M.; Haddadin, S. Collision detection, isolation and identification for humanoids. In Proceedings of the 2017 IEEE International Conference on Robotics and Automation (ICRA), Las Vegas, NV, USA, 29 May–3 June 2017; pp. 4754–4761.
15. Stasse, O.; Escande, A.; Mansard, N.; Miossec, S.; Evrard, P.; Kheddar, A. Real-time (self)-collision avoidance task on a hrp-2 humanoid robot. In Proceedings of the 2008 IEEE International Conference on Robotics and Automation (ICRA), Pasadena, CA, USA, 19–23 May 2008; pp. 3200–3205.
16. Fang, C.; Rocchi, A.; Hoffman, E.M.; Tsagarakis, N.G.; Caldwell, D.G. Efficient self-collision avoidance based on focus of interest for humanoid robots. In Proceedings of the 2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids), Seoul, Korea, 3–5 November 2015; pp. 1060–1066.
17. Salehian, S.S.M.; Figueroa, N.; Billard, A. A unified framework for coordinated multi-arm motion planning. *Int. J. Robot. Res.* **2018**, *37*, 1205–1232. [[CrossRef](#)]
18. Dietrich, A.; Wimböck, T.; Täubig, H.; Albu-Schaffer, A.; Hirzinger, G. Extensions to reactive self-collision avoidance for torque and position controlled humanoids. In Proceedings of the 2011 IEEE International Conference on Robotics and Automation, Shanghai, China, 9–13 May 2011; pp. 3455–3462. [[CrossRef](#)]
19. Sugiura, H.; Gienger, M.; Janssen, H.; Goerick, C. Real-Time Self Collision Avoidance for Humanoids by means of Nullspace Criteria and Task Intervals. In Proceedings of the 2006 6th IEEE-RAS International Conference on Humanoid Robots, Genova, Italy, 4–6 December 2006; pp. 575–580. [[CrossRef](#)]
20. Sugiura, H.; Gienger, M.; Janssen, H.; Goerick, C. Real-time collision avoidance with whole body motion control for humanoid robots. In Proceedings of the 2007 IEEE/RSJ International Conference on Intelligent Robots and Systems, San Diego, CA, USA, 29 October–2 November 2007; pp. 2053–2058. [[CrossRef](#)]
21. Kuffner, J.J.; LaValle, S.M. RRT-connect: An efficient approach to single-query path planning. In Proceedings of the 2000 ICRA, Millennium Conference, IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065), San Francisco, CA, USA, 24–28 April 2000; pp. 995–1001. [[CrossRef](#)]
22. Vahrenkamp, N.; Asfour, T.; Dillmann, R. Simultaneous Grasp and Motion Planning: Humanoid Robot ARMAR-III. *IEEE Robot. Autom. Mag.* **2012**, *19*, 43–57. [[CrossRef](#)]
23. Kalakrishnan, M.; Chitta, S.; Theodorou, E.; Pastor, P.; Schaal, S. STOMP: Stochastic trajectory optimization for motion planning. In Proceedings of the 2011 IEEE International Conference on Robotics and Automation, Shanghai, China, 9–13 May 2011; pp. 4569–4574. [[CrossRef](#)]
24. Pavlichenko, D.; Behnke, S. Efficient stochastic multicriteria arm trajectory optimization. In Proceedings of the 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Vancouver, BC, Canada, 24–28 September 2017; pp. 4018–4025. [[CrossRef](#)]
25. Kabir, A.M.; Kanyuck, A.; Malhan, R.K.; Shembekar, A.V.; Thakar, S.; Shah, B.C.; Gupta, S.K. Generation of Synchronized Configuration Space Trajectories of Multi-Robot Systems. In Proceedings of the 2019 International Conference on Robotics and Automation (ICRA), Montreal, QC, Canada, 20–24 May 2019; pp. 8683–8690.
26. How Multi Robot Can Avoid the Collision. Available online: <https://forums.robotstudio.com/discussion/10439/how-multi-robot-can-avoid-the-collision> (accessed on 24 August 2020).
27. Flacco, F.; Kroeger, T.; De Luca, A.; Khatib, O. A Depth Space Approach for Evaluating Distance to Objects. *J. Intell. Robot. Syst.* **2014**, *80*, 7–22. [[CrossRef](#)]
28. Rimon, E.; Koditschek, D.E. The construction of analytic diffeomorphisms for exact robot navigation on star worlds. *Trans. Am. Math. Soc.* **1991**, *327*, 71–116. [[CrossRef](#)]