Robotica (2017) volume 35, pp. 1431–1450. © Cambridge University Press 2016
doi:10.1017/S0263574716000205

Collision-free motion planning of a virtual arm based on the FABRIK algorithm

Songqiao Tao[†]*, Yumeng Yang[‡]

[†]*Department of Mechanic and Electronic Engineering, Wuhan Technical College of Communications, 6 Baishazhou Road, Wuhan City, Hubei Province, 430065, P. R. China*
[‡]*School of Mechanical Science and Engineering, Huazhong University of Science and Technology, 1037 Luoyu Road, Wuhan City, Hubei Province, 430074, P. R. China*
Email: yym20071202@163.com

(Accepted March 14, 2016. First published online: April 18, 2016)

SUMMARY
Collision-free motion planning of a virtual arm is an intractable task in high-interference environments. In this paper, an approach for collision-free motion planning of a virtual arm based on the forward and backward reaching inverse kinematics (FABRIK) algorithm is proposed. First, a random rotation strategy and local optimum-seeking technology are introduced to improve the FABRIK algorithm in order to avoid obstacles. The improvement FABRIK algorithm is used to design the final grasping posture of a virtual arm according the target position. Then, a bidirectional rapidly exploring random tree (Bi-RRT) algorithm is adopted to explore the process postures from a given initial posture to the final grasping posture. Different from the existing method, the proposed Bi-RRT algorithm in this paper plans the motions of a virtual arm in a seven-dimensional angle space, and the final grasping posture is automatically designed using the obstacle-avoidance FABRIK algorithm instead of the manual design. Finally, the post-processing technique is introduced to remove redundant nodes from the planned motions. This procedure has resolved the problem that the Bi-RRT algorithm is a random algorithm. The experimental results show the proposed method for collision-free motion planning of a virtual arm is feasible.

KEYWORDS: Collision-free; Motion planning; Inverse kinematics; Obstacle avoidance; Virtual arm.

1. Introduction

In recent years, the technologies of a virtual human have received extensive attentions in the fields of virtual maintenance, virtual assembly and computer animation. As the work environment is complex, a virtual human is prone to interfere with its surrounding scenarios. For virtual maintenance and assembly operations, the basic tasks are to move the part, grasp the tool, screw the nut and press the button, and these actions are performed by the movements of upper limbs. Meanwhile, the grab of an arm is one of the basic movements of upper limbs. Then, the motion planning of a virtual arm in high-interference environments has been an active topic. For the collision-free grab of a virtual arm, we need to design and control its postures.

The typical technologies for designing and controlling the postures of a virtual human have motion capture technique, forward kinematics (FK) and inverse kinematics (IK). Motion capture is the process of recording the movement of objects or people. But often, specific hardware and special software programs are required to obtain and process the data. The FK refers to the use of the kinematics equations of a robot to compute the position of the end-effector from specified values for the joint parameters. As the FK usually need to adjust each joint of a virtual arm manually, it is complex and counterintuitive in high-interference environments. The IK refers to the use of the kinematics equations of a robot to determine the joint parameters that provide a desired position of the end-effector. As the IK reverses the configuration of the chain to determine the joint parameters

* Corresponding author. E-mail: taosongqiao@163.com

https://doi.org/10.1017/S0263574716000205 Published online by Cambridge University Press

that achieves a desired configuration, it has been widely used to design and control the postures of a virtual human.

For the grasping action of a virtual arm, the IK algorithm finds its solution according to the target position. Unfortunately, the IK cannot find process postures between the initial position and target position. In most cases, the simulation software with the interpolation function of pose parameters is adopted to resolve this problem. However, the generated process postures may interfere with their surrounding scenarios. Thus, we introduce the motion planning to design and control process postures. To avoid obstacles, the combination of an IK algorithm and motion planning is a great choice.

In this paper, a method for collision-free motion planning of a virtual arm based on the FABRIK algorithm is presented. First, a FABRIK algorithm with obstacle-avoidance is proposed to design the final collision-free grasping posture of a virtual arm according to the target position. The FABRIK algorithm[1,2] is an IK algorithm with good convergence and low computational cost. To avoid obstacles, the joint constraints are added and the surrounding scenarios are considered for solving the IK problem. Then, a Bi-RRT algorithm is introduced to explore the process postures of a virtual arm from a given initial posture to the final grasping posture. The Bi-RRT algorithm is very efficient for solving single-query path planning problems in high-dimensional configuration spaces.[3,4] Finally, some strategies are adopted to handle bound constraints of an arm's 7 angular parameters and to remove redundant nodes from the planned motions. The experimental results show the proposed method can achieve collision-free grasping actions of a virtual arm in high-interference environments.

The rest of this paper is organized as follows. First, literature reviews are addressed in Section 2. Then, an improved FABRIK algorithm for designing the grasping posture of a virtual arm is introduced in Section 3. In Section 4, the motion planning of a virtual arm based on the Bi-RRT algorithm is given in detail. Following this section, some experimental results based on the proposed method are given in Section 5. Finally, the paper presents some conclusions in Section 6.

2. Literature Reviews

2.1. Inverse kinematics

Most of the existing technologies for IK problems can be classified into Jacobian-based, neural network-based and heuristic-based methods.

Jacobian-based methods use a Jacobian matrix to represent the linear approximation of the non-linear function for an IK problem. To calculate or approximate the inverse of a Jacobian matrix, some methodologies have been proposed, such as the Jacobian transpose,[5,6] singular value decomposition,[7,8] damped least squares[9,10] and incorporating constraints.[11,12] Jacobian-based approaches usually find smooth postures. However, most of them suffer from the problems of high computational cost and the singularity.

To reduce the computational cost, artificial neural networks are introduced to solve IK problems.[13–18] Tejomurtula *et al.*[13] presented an IK solution for a three-joint robotic manipulator based on neural-networks. In their work, structured neural networks effectively improve the training time and accuracy of back propagation algorithms. Karlık *et al.*[14] proposed an IK solution for 6-DOF robot manipulator based on identifying the best neural network configuration. They thought that the solution of six separate neural networks was superior to a single neural network with six outputs. Mayorga *et al.*[15] proposed a neural network method for fast IK computation. To avoid the singularities of a planar redundant manipulator, the bounded geometrical concepts were adopted to establish characterizing matrices. To minimize the error at the end effector and obtain more precise solutions, Köker[16] adopted neural network and genetic algorithm to solve the IK problem of a six-joint robot. The genetic algorithm effectively enhances the precision of neural network results in the floating-point portion. Köker *et al.*[17] proposed a neural network committee machine for solving the IK problem of a 6-DOF robot manipulator. In their work, 10 parallel neural networks formed a committee machine, and the solution of each neural network was evaluated for finding the best prediction result.

Heuristic-based method is the simplest and fastest IK solver.[2] The typical heuristic-based methods have Cyclic Coordinate Descent (CCD)[19–22] and Triangulation IK.[23,24] CCD approach for solving IK problems is firstly introduced in ref. [19]. CCD[20–22] usually has a low computational cost in an iteration and can handle the IK problem without matrix manipulations. However, CCD may

https://doi.org/10.1017/S0263574716000205 Published online by Cambridge University Press

produce unrealistic postures and it is difficult to handle the problems with multiple end effectors. The triangulation IK[23,24] uses the cosine law to calculate each joint angle; it can reach the target position in few iterations and has low computational cost. However, it is difficult to handle the problems with multiple end effectors.

Aristidou *et al*.[1,2] proposed a new heuristic approach called FABRIK for solving the IK problem. The FABRIK algorithm is a heuristic iterative approach for solving IK problems in different scenarios. It uses a forward and backward iterative mode to make the end-effector approach the target position quickly. As the FABRIK algorithm explores each joint position by locating a point on a line instead of using angle rotations, the algorithm has low computational complexity. Moreover, the FABRIK algorithm can solve the IK problem with multiple targets. However, the surrounding scenario has not yet been considered for solving the IK problem, the solution of the IK problem is easy to interfere with its surrounding scenario. To resolve this problem, a random rotation strategy and local optimum-seeking technology are introduced to improve the FABRIK algorithm in this paper. And the improvement FABRIK algorithm is used to design the final grasping posture of a virtual arm according the target position.

2.2. Motion planning

Motion planning is a term used in robotics for the process of breaking down a desired movement task into discrete motions that satisfy movement constraints and possibly optimize some aspect of the movement. Exact motion planning for high-dimensional systems under complex constraints is intractable computationally. Most of the motion planning for high-dimensional systems can be classified into Artificial Potential Field (APF) algorithm[25–32] and Sampling-based algorithm.[33–41]

The APF algorithm was firstly presented for manipulators in ref. [25]. As the APF algorithm is simple and high efficient, it is an attractive method for the motion planning. However, the APF algorithm has several significant inherent limitations. In ref. [26], Koren *et al*. introduced the virtual force field method to identify the shortcomings of local minima, non-smooth movement, oscillations in the presence of obstacles and the oscillation in a narrow passage. Ren *et al*.[27] presented a modified Newton's method to overcome the inherent oscillation problems of APF in the presence of obstacles and in narrow passages. In ref. [28], the equipotential line method was proposed to escape the local minima for the APF algorithm. Jia *et al*.[29] proposed a new attractive function for the APF algorithm, which can reduce the potential around goals evidently to eliminate the problem of goals non-reachable with obstacles nearby. To deal with two potential risks of a robot-obstacle collision and local minima trap, Lee *et al*.[30] proposed new potential functions with Random force algorithms for APF. In ref. [31], Velagic *et al*. proposed a modified APF with fuzzy rules and suitable steps to pull the robot out of the local minima. Raja *et al*.[32] introduced a gradient function in the conventional APF approach to plan motion on the rough three-dimensional terrain. As four different cost functions respectively give the path constraints in relation to energy, traction force, slip and deviation from a straight line, the new APF eliminates oscillations and avoids concave untraversable area without getting trapped in it.

As sampling-based algorithms avoid the explicit construction of obstacles in a state space, a large amount of computation time is saved. Probabilistic RoadMaps (PRMs)[33,34] and rapidly exploring random trees (RRTs)[35,36] are the most influential sampling-based algorithms for the motion planning. They both randomly sample the connecting points from the state space, but they construct a graph connecting the points in different ways. To improve future planners, Hsu *et al*.[37] introduced probabilistic foundations of the PRM planning to fill the gap and identify promising directions. Rantanen *et al*.[38] proposed a configuration deactivation method to improve the execution time used to build the roadmap in PRMs planners. However, PRMs are unable to determine that no path exists, and they have a probability of failure that decreases to zero as more time is spent.

RRT is an algorithm designed to efficiently search non-convex, high-dimensional spaces by randomly building a space-filling tree. The tree is constructed incrementally from samples drawn randomly and the search space, and is inherently biased to grow towards large unsearched areas of the problem.[3] The RRT algorithm easily handles the problems with obstacles and differential constraints. Kothari *et al*.[39] proposed a probabilistically robust path planning method for unmanned aerial vehicles in the presence of uncertainty using the RRT. A chance constraint is adopted to limit the probability of constraint violation and extend this framework to deal with uncertain dynamic obstacles. Inspired by classical bidirectional search ideas, the updated RRT algorithm called Bi-RRT has appeared. The Bi-RRT algorithm grows two RRTs, one rooted at the initial state and another

https://doi.org/10.1017/S0263574716000205 Published online by Cambridge University Press

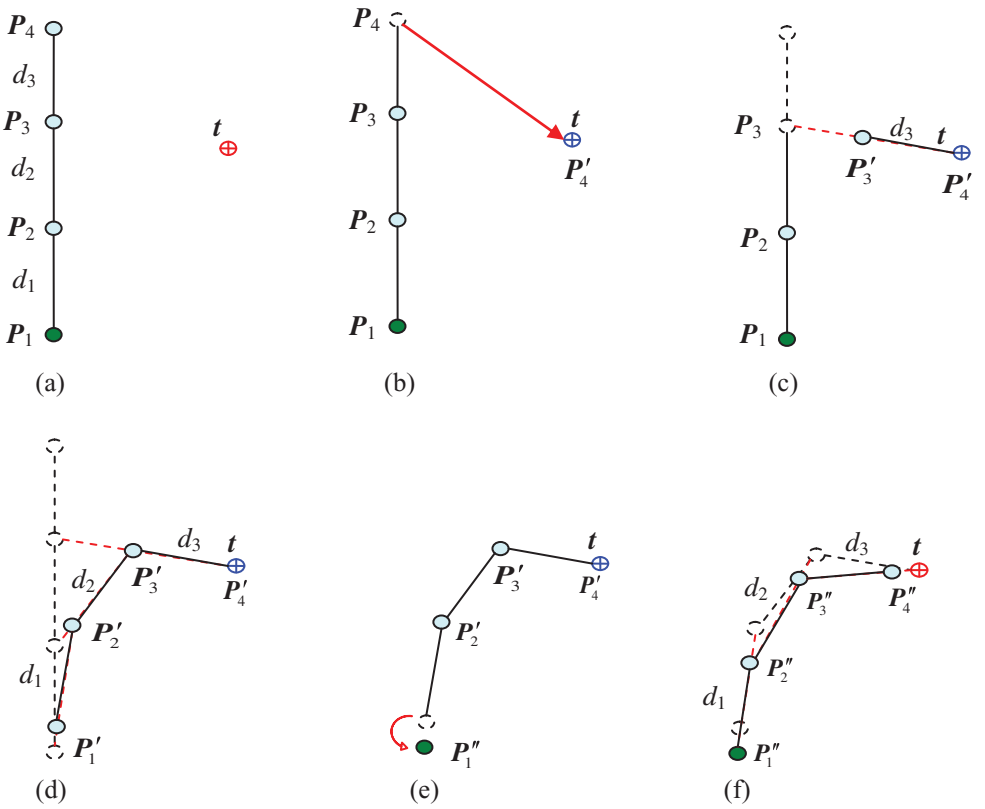


Fig. 1. An example of a full iteration of the FABRIK for a single target and four manipulator joints. (a) The initial position of the manipulator and the target. (b) The end effector P_4 is moved to the target position t. (c) Find the updated joint position P_3'. (d) Find updated joint positions P_1' and P_2'. (e) The root joint P_1' is moved to its initial position P_1. (f) Find updated joint positions P_2'', P_3'' and P_4''.

rooted at the target state. The algorithm searches for the states that are "common" to both trees.[40] Compared with RRT algorithm, Bi-RRT algorithm has higher efficiency. Qureshi *et al.*[41] proposed an intelligent Bi-RRT method for optimal motion planning in complex cluttered environments. A variant of the intelligent Bi-RRT called the Bi-RRT* has been introduced to ensure the asymptotic optimality. As the obstacle-avoidance FABRIK algorithm cannot find process postures between the initial position and target position, we introduce the Bi-RRT algorithm to explore the process postures of a virtual arm from a given initial posture to the final grasping posture in this paper.

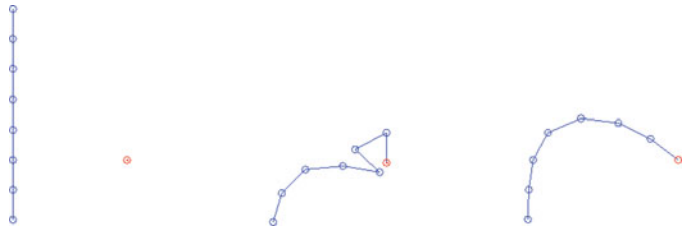
3. An Improved FABRIK Algorithm

3.1. FABRIK algorithm

An example of a full iteration of the FABRIK algorithm for a single target and four manipulator joints is listed in Fig. 1. Let P_i ($i = 1\sim4$) be the joint position of a manipulator, and P_1 be the root joint and P_4 be the end effector. For the simplification, only a single end effector exists. Let t be the target position and d_j ($j = 1\sim3$) be the distance between two adjacent joints. A full iteration of the FABRIK algorithm can be expressed as follows:

First, the distance $d_j = |P_{j+1}\text{-}P_j|$ ($j = 1\sim3$) is calculated. Then, check whether the target position is reachable or not. If $|d_1 + d_2 + d_3| \geq |t - P_1|$, the target position is within reach. Otherwise, it is unreachable. If the target position is within reach, a full iteration is performed. The initial position of the manipulator and the target position are given in Fig. 1(a) and (b)–(d) represent the forward search while Fig. 1(e)–(f) represent the backward search. In Fig. 1(b), the end effector P_4 is moved to the target position t, $P_4' = t$. In Fig. 1(c), find the updated joint position P_3' which lies on the line l_3 that passes through the points P_4' and P_3, and $|P_4' - P_3'| = d_3 = |P_4 - P_3|$. For the joints P_1 and P_2, the same method is adopted to find updated joint positions P_1' and P_2' (see Fig. 1 (d)). In Fig. 1(e), the

https://doi.org/10.1017/S0263574716000205 Published online by Cambridge University Press



(a) (b) (c)

Fig. 2. An example for solving the IK problem with a single target and eight manipulator joints. (a) The initial position of the manipulator and the target. (b)The solution for the CCD algorithm. (c) The solution for the FABRIK algorithm.

root joint P_1' is moved to its initial position P_1, $P_1'' = P_1$. In Fig. 1(f), start from the root joint P_1 and move outwards to the end effector t, repeat the same procedure to find updated joint positions P_2'', P_3'' and P_4''. Compared with the initial distance between the end effector P_4 and the target position t, the distance between P_4'' and t is obviously reduced. The algorithm is repeated until the distance between the end effector and the target position is less than the convergence precision ε.

To validate the FABRIK algorithm, we use Matlab to solve the IK problem with a single target and eight manipulator joints. The solutions of the IK problem are given in Fig. 2. In Fig. 2, the blue parts represent the joint chain while the red circle describes the target position. Besides, the initial position of the manipulator and the target are given in Fig. 2(a) while the solutions for the CCD algorithm and FABRIK algorithm are respectively listed in Fig. 2(b) and (c). For the IK problem with a single target and eight manipulator joints, the CCD algorithm and FABRIK algorithm have approximate computational efficiency. As you can see from Fig. 2(b) and (c), the solution of the FABRIK algorithm for the IK problem has visually smooth postures. The reason is that the sum of joint rotation angles is smaller in the FABRIK algorithm.

Then, we use the FABRIK algorithm to solve the IK problem with multiple targets. The solutions for the IK problem with two and five end effectors are listed in Fig. 3. In Fig. 3(a) and (b), the diagram in the left represents the initial position of the manipulator and the target while the diagram in the right represents the solution of the FABRIK algorithm. As you can see from Fig. 3(a) and (b), the solutions of the FABRIK algorithm for the IK problem with two and five end effectors have visually smooth postures. The experimental results show the FABRIK algorithm can solve the IK problem with multiple targets.

3.2. Obstacle-avoidance FABRIK algorithm

Most of IK algorithms for solving IK problems have not considered the effect of surrounding scenarios. Here, the surrounding scenarios are considered for the improved FABRIK algorithm. To avoid obstacles, a random rotation strategy and local optimum-seeking technology are introduced in each iteration.

The Random rotation strategy refers to the case that if a connecting rod of a joint chain interferes with its surrounding scenarios, the connecting rod will randomly take a rotation in degrees for avoiding obstacles. By taking a rotation, the connecting rod will have an updated position. As the rotation is random, the updated position of the connecting rod might be not the global optimal one. Since the FABRIK algorithm is a heuristic iterative approach, subsequent iterations will gradually improve the previous results, and the optimal result can be found in the end. Based on our experiments, we find that the correct position can be obtained by taking a random rotation of a connecting rod in most cases. But the algorithm might have no convergence in finite rotations. To resolve this problem, we randomly take many rotations and select the optimization result from multiple results as the ultimate result.

The local optimum-seeking technology is that only a joint is optimized in an iteration. To ensure the joint chain without major changes, the sampling point closest to the root joint is chosen as the local optimal solution in the forward search. Meanwhile, to make the end effector approach the target position as far as possible, the sampling point closest to the target position is chosen as the local optimal solution in the backward search. Besides, too many sampling points will affect the iteration

https://doi.org/10.1017/S0263574716000205 Published online by Cambridge University Press

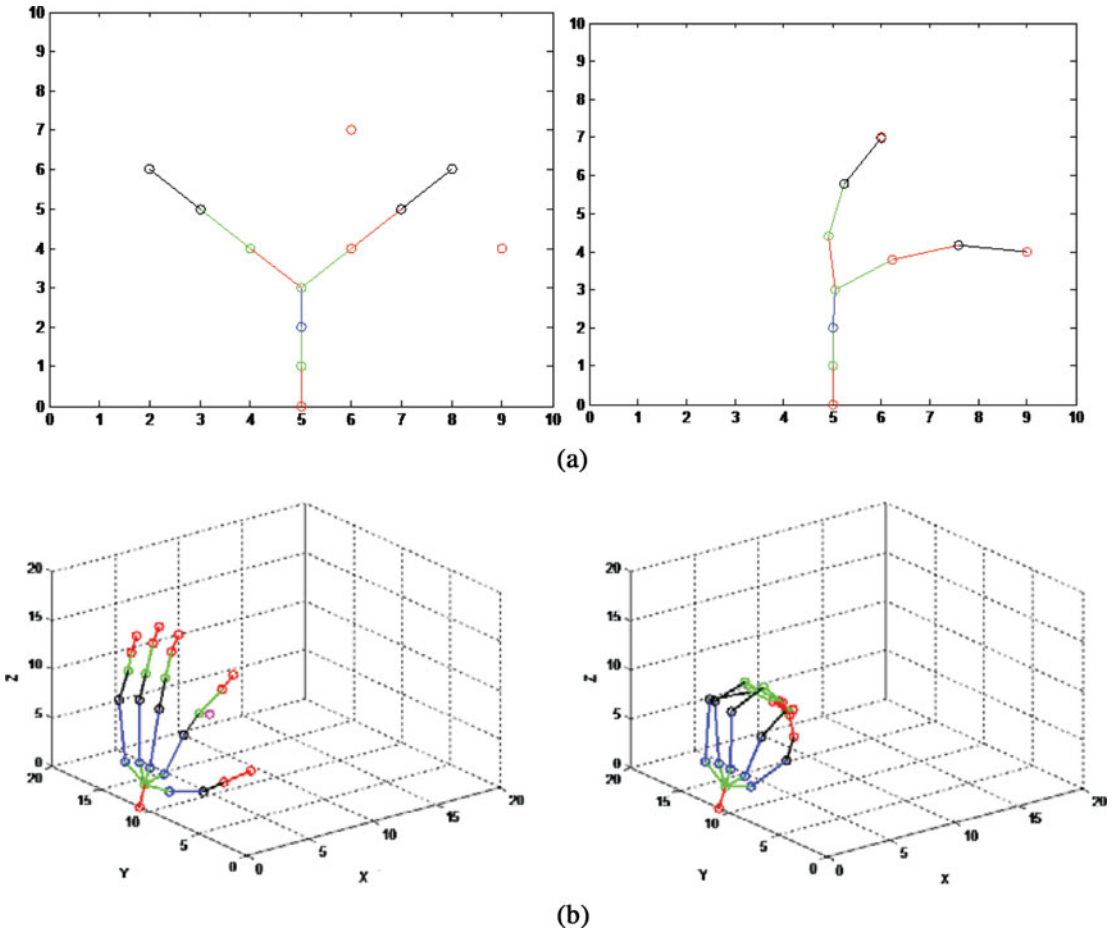


Fig. 3. An example for solving the IK problem with multiple end effectors. (a) The solution of the FABRIK algorithm for the IK problem with two end effectors. (b) The solution of the FABRIK algorithm for the IK problem with five end effectors.

efficiency and too few sampling points will not ensure the diversity of the sample. In general, the sampling number is determined by the structure of the joint chain and its surrounding scenarios. For planar joint chains, about 5∼10 sampling points can ensure that the success rate of the algorithm is more than 95%. But for spatial joint chains, the same success rate for the algorithm is needed to sample 10∼15 points.

To improve the efficiency of the interference checking, the bounding box technology is introduced. First, the obstacles are replaced by their corresponding sphere bounding boxes. Then, the interpolation calculation is performed for each connecting rod of a joint chain, and the minimum distance between the interpolation point and the center of the spherical bounding box is measured. Finally, the minimum distance is compared with the radius of bounding box. If the minimum distance is larger than the radius of bounding box, the interference exists; otherwise, the interference does not exist.

An example for illustrating the FABRIK algorithm with a random rotation strategy and local optimum-seeking technology is listed in Fig. 4. Let P_i $(i = 1\sim4)$ be the joint position of a manipulator, and P_1 be the root joint and P_4 be the end effector, and d_j $(j = 1\sim3)$ be the distance between the two adjacent joints, and t be the target position. The shadows represent bounding boxes of the obstacles in Fig. 4. Figure 4(a) gives the initial position of a manipulator, the positions of the target and obstacle. In Fig. 4(b), the end effector P_4 is moved to the target position t, $P_4' = t$, and find the updated joint position P_3^0 which lies on the line l_3 that passes through the points P_4' and P_3, and $|P_4' - P_3^0| = d_3$. Obviously, the obstacle will interfere with the connecting rod $P_3^0 P_4'$. Then, a random rotation strategy is adopted to make the connecting rod $P_3^0 P_4'$ rotate around the joint position P_4', until the obstacle doesn't interfere with the connecting rod. Taking multiple rotations for the connecting

https://doi.org/10.1017/S0263574716000205 Published online by Cambridge University Press

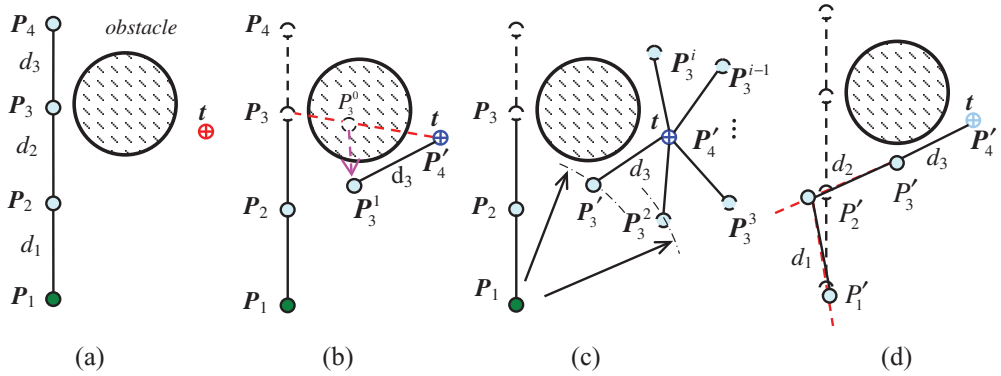


Fig. 4. An example for illustrating the FABRIK algorithm with a random rotation strategy and local optimum-seeking technology. (a) The initial position. (b) Rotate $P_3'P_4'$ for avoiding the obstacle. (c) Take multiple rotations for $P_3'P_4'$. (d) Find updated joint positions P_1' and P_2'.

rod $P_3^0P_4'$ around P_4', multiple new positions $P_3{}^k$ ($k = 1, 2,$) of P_3^0 without the interference can be found. Thus, the optimal joint position P_3' is selected from $P_3{}^k$, which has the minimum distance with the root join P_1 (see Fig. 4(c)). For the joints P_1 and P_2, the same method is adopted to find updated joint positions P_1' and P_2' (see Fig. 4(d)). Figure 4(b)–(d) represent collision-free forward search. For the backward search, start from the root joint P_1 and move outwards to the end effector, repeat the same procedure to find updated joint positions P_2'', P_3'' and P_4''. The detailed procedures of the FABRIK algorithm with a random rotation strategy and local optimum-seeking technology are presented as follows:

Input: The joint position P_i ($i = 1, 2, n$), the target position t, the distance d_j between the two adjacent joints P_j and P_{j+1}, and $d_j = |P_{j+1} - P_j|$ ($j = 1, 2, n-1$), the convergent precision ε, the maximum number $\boldsymbol{n}_m$ of iterations.

Output: The updated collision-free joint position $P_i{}''$ ($i = 1, 2, n$).

Step 1: Check whether the target is reachable or not:

If $|t - P_1| > |d_1 + d_2 + \ldots\ldots + d_{n-1}|$, the target position is unreachable, go to Step 5; otherwise, it is within reach, go to the next step.

Step 2: Collision-free forward search:

Step 2.1. Move the end effector P_n to the target position t, $P_n{}' = t$, and find the updated joint position P_{n-1}^0 which lies on the line l_{n-1} that passes through the points $P_n{}'$ and P_{n-1}, and $|P_n{}' - P_{n-1}^0| = d_n$;

Step 2.2. Check whether the obstacle will interfere with the connecting rod $P_{n-1}^0P_n{}'$ or not:

If the interference doesn't exist, $P_{n-1}{}' = P_{n-1}^0$ go to Step 2.5; otherwise, go to the next step;

Step 2.3. Take multiple rotations for $P_{n-1}^0P_n{}'$ around $P_n{}'$, find multiple updated positions P_{n-1}^k (k =1, 2,) of P_{n-1}^0 without the interference;

Step 2.4. Select the optimal joint position $P_{n-1}{}'$ from P_{n-1}^k, which has the minimum distance with the root join P_1;

Step 2.5. Repeat the procedures Step 2.1 ~ Step 2.4, until the optimal root joint position P_1' is found.

Step 3: Collision-free backward search:

Step 3.1. Move P_1' to the initial position of the root joint P_1, and find the updated joint position P_1'' which lies on the line l_1 that passes through the points P_1'' and P_2', and $|P_2' - P_1''| = d_1$;

Step 3.2. Check whether the obstacle will interfere with the connecting rod $P_1''P_2'$ or not:

https://doi.org/10.1017/S0263574716000205 Published online by Cambridge University Press

Table I. The experimental results of the obstacle-avoidance FABRIK algorithm.

Serial number	Whether the target is reachable or not	The number of iterations	The distance between the end effector and the target position(mm)	Run time (s)	Experimental results
1	Yes	4	2.25×10^{-4}	0.0411	Fig. 5(a)
2	Yes	6	1.95×10^{-4}	0.1196	Fig. 5(b)
3	Yes	57	5.78×10^{-4}	0.8269	Fig. 5(c)
4	Yes	3	6.94×10^{-4}	0.0601	Fig. 5(d)
5	Yes	6	2.31×10^{-4}	0.0975	Fig. 5(e)
6	Yes	2	7.79×10^{-4}	0.0478	Fig. 5(f)
7	No	100	0.55	3.0460	Fig. 5(g)
8	Yes	3	2.08×10^{-4}	0.0846	Fig. 5(h)
9	Yes	4	6.59×10^{-4}	0.0969	Fig. 5(i)
10	No	100	0.31	1.8944	Fig. 5(j)

If the interference doesn't exist, go to Step 3.5; otherwise, go to the next step;

Step 3.3. Take multiple rotations for $P_1''P_2'$ around P_1'', find multiple updated collision-free positions $P_2''^k$ (k =1, 2,) of P_2';

Step 3.4. Select the optimal joint position P_2'' from $P_2''^k$, which has the minimum distance with the target t;

Step 3.5. Repeat the procedures Step 3.1 $\sim$ Step 3.4, until the optimal end effector position P_n'' is found.

Step 4: Check whether the obstacles will interfere with connecting rods $P_1''P_2''$, $P_2''P_3''$, ..., $P_{n-1}''P_n''$, and the distance between the end effector P_n'' and the target position t is less than the convergent precision ϵ or not:

If both conditions are satisfied, the target position is reachable, output the updated collision-free joint position P_i'' (i =1, 2, n); otherwise, go to Step 2. If the number of iterations is more than the maximum number of iterations, and the two conditions cannot be satisfied, the target position is unreachable.

To validate the obstacle-avoidance FABRIK algorithm, we use Matlab to solve the IK problem with four manipulator joints. The joint coordinates are respectively (0, 0), (0, 2), (0, 4) and (0, 6), and the target position is (5, 5). Five rounds are used for representing obstacles, their radius are randomly generated between 0 and 1. And the centers of five rounds are randomly set in a rectangle with diagonal vertices (0.5, 0.5) and (4.5, 4.5). The interference criterion is that whether the distance between the joint and the center of the round is less than the radius of the round. The number of sampling points is set to 10, and the number of iterations is no more than 100, and the convergence precision ϵ is within 0.001. The experimental results of 10 trials are listed in Table I and Fig. 5. In Fig. 5, the blue parts, red circles and pink rounds respectively represent the joint chain, target position and the obstacles. The joint chains remained upright represent the initial position while other joint chains represent the final grasping position. For 10 trials with various size and position of obstacles, eight times are reachable (see Fig. 5(a)–(f), (h), (i)) while two times are unreachable (see Fig. 5(g) and (j)), and the number of iterations is usually no more than 10 (serial number 1, 2, 4, 5, 6, 8 and 9 in Table I). The experimental results show that the FABRIK algorithm has strong adaptability and fast convergence rate. Even if the target position t is unreachable in Fig. 5(g) and (j), the end effector is the closest thing to the target position.

3.3. Obstacle-avoidance FABRIK algorithm with constraints

In consideration of natural physiological conditions, the rotation angle of human joints should be limited in a certain range. Thus, the rotation angle of a joint has a range in solving the IK problem,

https://doi.org/10.1017/S0263574716000205 Published online by Cambridge University Press

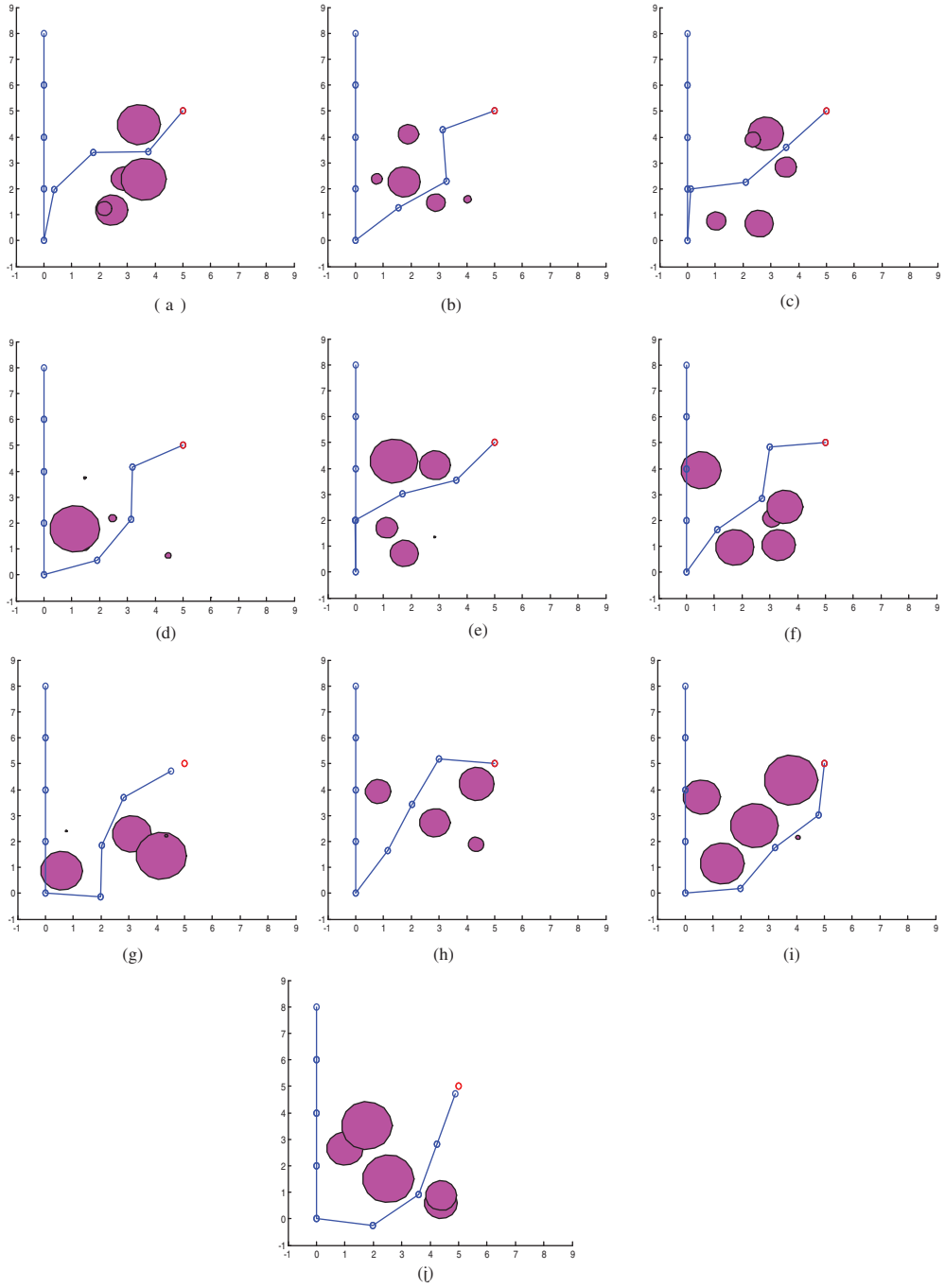


Fig. 5. The experimental results of the obstacle-avoidance FABRIK algorithm.

and this angle should be calculated for adding the constraint of a joint. As the FABRIK algorithm has found the joint position, the analytical method can be adopted to get the rotation angle of a joint from the found joint position. For the convenience of calculation, a simplified model for a human arm is proposed in Fig. 6. The components of an arm have upper arm, lower arm and palm, which correspond to shoulder joint, elbow joint and wrist joint. In Fig. 6, α_1, α_2 and α_3 respectively represent three rotational degrees of freedom for a shoulder joint, and α_4 represents rotation degrees of freedom for an elbow joint while α_5, α_6 and α_7 respectively represent three rotation degrees of freedom for a wrist joint.

https://doi.org/10.1017/S0263574716000205 Published online by Cambridge University Press

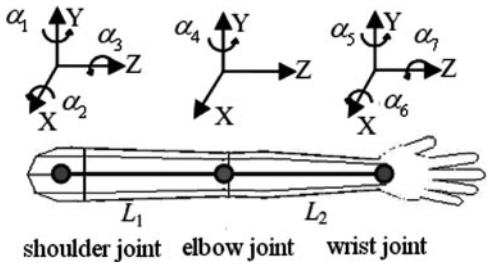


Fig. 6. The simplified model of a human arm.

The transform matrices of each joint can be expressed as follows:

$$A_1 = R_y(\alpha_1) = \begin{bmatrix} \cos\alpha_1 & 0 & \sin\alpha_1 & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\alpha_1 & 0 & \cos\alpha_1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}; \quad A_2 = R_x(\alpha_2) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha_2 & -\sin\alpha_2 & 0 \\ 0 & \sin\alpha_2 & \cos\alpha_2 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix};$$

$$A_3 = R_z(\alpha_3)T(0, 0, -L_1) = \begin{bmatrix} \cos\alpha_3 & -\sin\alpha_3 & 0 & 0 \\ \sin\alpha_3 & \cos\alpha_3 & 0 & 0 \\ 0 & 0 & 1 & -L_1 \\ 0 & 0 & 0 & 1 \end{bmatrix};$$

$$A_4 = R_y(\alpha_4)T(0, 0, -L_2) = \begin{bmatrix} \cos\alpha_4 & 0 & \sin\alpha_4 & -L_2\sin\alpha_4 \\ 0 & 1 & 0 & 0 \\ -\sin\alpha_4 & 0 & \cos\alpha_4 & -L_2\cos\alpha_4 \\ 0 & 0 & 0 & 1 \end{bmatrix};$$

$$A_5 = R_y(\alpha_5) = \begin{bmatrix} \cos\alpha_5 & 0 & \sin\alpha_5 & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\alpha_5 & 0 & \cos\alpha_5 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}; A_6 = R_x(\alpha_6) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha_6 & -\sin\alpha_6 & 0 \\ 0 & \sin\alpha_6 & \cos\alpha_6 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix};$$

$$A_7 = R_z(\alpha_7) = \begin{bmatrix} \cos\alpha_7 & -\sin\alpha_7 & 0 & 0 \\ \sin\alpha_7 & \cos\alpha_7 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

Let A_w be the position matrix of a wrist joint relative to a shoulder joint. A_w can be expressed as follows:

$$A_w = \begin{bmatrix} r_{11} & r_{12} & r_{13} & w_x \\ r_{21} & r_{22} & r_{23} & w_y \\ r_{31} & r_{32} & r_{33} & w_z \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

Here, we have

$$\begin{aligned} r_{11} = {} & [(\cos\alpha_1\cos\alpha_3 + \sin\alpha_1\sin\alpha_2\sin\alpha_3)\cos\alpha_4 - \sin\alpha_1\cos\alpha_2\sin\alpha_4](\cos\alpha_5\cos\alpha_7 + \sin\alpha_5\sin\alpha_6\sin\alpha_7) \\ & + (\sin\alpha_1\sin\alpha_2\cos\alpha_3 - \cos\alpha_1\sin\alpha_3)(\cos\alpha_6 + \sin\alpha_7) + [(\cos\alpha_1\cos\alpha_3 + \sin\alpha_1\sin\alpha_2\sin\alpha_3)\sin\alpha_4 \\ & + \sin\alpha_1\cos\alpha_2\cos\alpha_4](-\sin\alpha_5\cos\alpha_7 + \cos\alpha_5\sin\alpha_6\sin\alpha_7) \\ r_{12} = {} & [(\cos\alpha_1\cos\alpha_3 + \sin\alpha_1\sin\alpha_2\sin\alpha_3)\cos\alpha_4 - \sin\alpha_1\cos\alpha_2\sin\alpha_4](-\cos\alpha_5\sin\alpha_7 + \sin\alpha_5\sin\alpha_6\cos\alpha_7) \\ & + (\sin\alpha_1\sin\alpha_2\cos\alpha_3 - \cos\alpha_1\sin\alpha_3)\cos\alpha_6\cos\alpha_7 + [(\cos\alpha_1\cos\alpha_3 + \sin\alpha_1\sin\alpha_2\sin\alpha_3)\sin\alpha_4 \\ & + \sin\alpha_1\cos\alpha_2\cos\alpha_4](-\sin\alpha_5\sin\alpha_7 + \cos\alpha_5\sin\alpha_6\cos\alpha_7) \end{aligned}$$

https://doi.org/10.1017/S0263574716000205 Published online by Cambridge University Press

$$
\begin{aligned}
r_{13} &= [(\cos\alpha_1\cos\alpha_3 + \sin\alpha_1\sin\alpha_2\sin\alpha_3)\cos\alpha_4 - \sin\alpha_1\cos\alpha_2\sin\alpha_4]\sin\alpha_5\cos\alpha_6 \\
&\quad - (\sin\alpha_1\sin\alpha_2\cos\alpha_3 - \cos\alpha_1\sin\alpha_3)\sin\alpha_6 \\
&\quad + [(\cos\alpha_1\cos\alpha_3 + \sin\alpha_1\sin\alpha_2\sin\alpha_3)\sin\alpha_4 + \sin\alpha_1\cos\alpha_2\cos\alpha_4]\cos\alpha_5\cos\alpha_6 \\
r_{21} &= [\cos\alpha_2\sin\alpha_3\cos\alpha_4 + \sin\alpha_2\sin\alpha_4](\cos\alpha_5\cos\alpha_7 + \sin\alpha_5\sin\alpha_6\sin\alpha_7) \\
&\quad + \cos\alpha_2\cos\alpha_3(\cos\alpha_6 + \sin\alpha_7) \\
&\quad + [\cos\alpha_2\sin\alpha_3\sin\alpha_4 - \sin\alpha_2\cos\alpha_4](-\sin\alpha_5\cos\alpha_7 + \cos\alpha_5\sin\alpha_6\sin\alpha_7) \\
r_{22} &= [\cos\alpha_2\sin\alpha_3\cos\alpha_4 + \sin\alpha_2\sin\alpha_4](-\cos\alpha_5\sin\alpha_7 + \sin\alpha_5\sin\alpha_6\cos\alpha_7) \\
&\quad + \cos\alpha_2\cos\alpha_3\cos\alpha_6\cos\alpha_7 \\
&\quad + [\cos\alpha_2\sin\alpha_3\sin\alpha_4 - \sin\alpha_2\cos\alpha_4](\sin\alpha_5\sin\alpha_7 + \cos\alpha_5\sin\alpha_6\cos\alpha_7) \\
r_{23} &= [\cos\alpha_2\sin\alpha_3\cos\alpha_4 + \sin\alpha_2\sin\alpha_4]\sin\alpha_5\cos\alpha_6 - \cos\alpha_2\cos\alpha_3\sin\alpha_6 \\
&\quad + [\cos\alpha_2\sin\alpha_3\sin\alpha_4 - \sin\alpha_2\cos\alpha_4]\cos\alpha_5\cos\alpha_6 \\
r_{31} &= [\cos\alpha_4(-\sin\alpha_1\cos\alpha_3 + \cos\alpha_1\sin\alpha_2\sin\alpha_3) - \cos\alpha_1\cos\alpha_2\sin\alpha_4](\cos\alpha_5\cos\alpha_7 + \sin\alpha_5\sin\alpha_6\sin\alpha_7) \\
&\quad + (\sin\alpha_1\sin\alpha_3 + \cos\alpha_1\sin\alpha_2\cos\alpha_3)(\cos\alpha_6 + \sin\alpha_7) + [\sin\alpha_4(-\sin\alpha_1\cos\alpha_3 \\
&\quad + \cos\alpha_1\sin\alpha_2\sin\alpha_3) + \cos\alpha_1\cos\alpha_2\cos\alpha_4](-\sin\alpha_5\cos\alpha_7 + \cos\alpha_5\sin\alpha_6\sin\alpha_7) \\
r_{32} &= [\cos\alpha_4(-\sin\alpha_1\cos\alpha_3 + \cos\alpha_1\sin\alpha_2\sin\alpha_3) - \cos\alpha_1\cos\alpha_2\sin\alpha_4](-\cos\alpha_5\sin\alpha_7 \\
&\quad + \sin\alpha_5\sin\alpha_6\cos\alpha_7) + (\sin\alpha_1\sin\alpha_3 + \cos\alpha_1\sin\alpha_2\cos\alpha_3)\cos\alpha_6\cos\alpha_7 - [\sin\alpha_4(-\sin\alpha_1\cos\alpha_3 \\
&\quad + \cos\alpha_1\sin\alpha_2\sin\alpha_3) + \cos\alpha_1\cos\alpha_2\cos\alpha_4](\sin\alpha_5\sin\alpha_7 + \cos\alpha_5\sin\alpha_6\cos\alpha_7) \\
r_{33} &= [\cos\alpha_4(-\sin\alpha_1\cos\alpha_3 + \cos\alpha_1\sin\alpha_2\sin\alpha_3) - \cos\alpha_1\cos\alpha_2\sin\alpha_4]\sin\alpha_5\cos\alpha_6 \\
&\quad - (\sin\alpha_1\sin\alpha_3 + \cos\alpha_1\sin\alpha_2\cos\alpha_3)\sin\alpha_6 \\
&\quad + [\sin\alpha_4(-\sin\alpha_1\cos\alpha_3 + \cos\alpha_1\sin\alpha_2\sin\alpha_3) + \cos\alpha_1\cos\alpha_2\cos\alpha_4]\cos\alpha_5\cos\alpha_6 \\
w_x &= -L_2[(\cos\alpha_1\cos\alpha_3 + \sin\alpha_1\sin\alpha_2\sin\alpha_3)\sin\alpha_4 + \sin\alpha_1\cos\alpha_2\cos\alpha_4] - L_1\sin\alpha_1\cos\alpha_2 \\
w_y &= -L_2[\cos\alpha_2\sin\alpha_3\sin\alpha_4 - \sin\alpha_2\cos\alpha_4] - L_1\sin\alpha_2 \\
w_z &= -L_2[\sin\alpha_4(-\sin\alpha_1\cos\alpha_3 + \cos\alpha_1\sin\alpha_2\sin\alpha_3) + \cos\alpha_1\cos\alpha_2\cos\alpha_4] - L_1\cos\alpha_1\cos\alpha_2
\end{aligned}
$$

Thus, $A_1A_2A_3A_4A_5A_6A_7 = A_w$.

For the obstacle-avoidance FABRIK algorithm, the position $e = [\,e_x\ e_y\ e_z\,]^T$ of a wrist joint relative to a shoulder joint can be found in a full iteration. Then, the analytical method for calculating the rotation angle of each joint can be expressed as follows:

Step 1: Set $A_1A_2A_3[\,0\ 0\ 0\ 1\,]^T = [\,e_x\ e_y\ e_z\ 1\,]^T$, calculate α_1 and α_2;

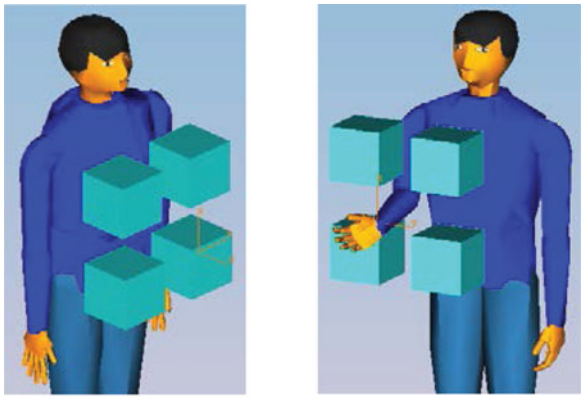
Step 2: As shoulder joint, elbow joint and wrist joint form a triangle, we can use triangle cosine theorem to calculate the rotation angle α_4 of an elbow joint;

Step 3: Set $A_3A_4[\,0\ 0\ 0\ 1\,]^T = A_2^{-1}A_1^{-1}[\,e_x\ e_y\ e_z\ 1\,]^T$, calculate α_3;

Step 4: Set $A_5A_6A_7 = A_4^{-1}A_3^{-1}A_2^{-1}A_1^{-1}A_w$, calculate α_5, α_6 and α_7.

In the following, the experiments have been conducted for the obstacle-avoidance grasping of a virtual arm. These tests were performed on a computer with an Intel 3.0 GHz CPU and 2.0 GB RAM, and the second developing tool (Tecnomatix/Process Simulate). The experimental subject is a joint chain of a right arm, and the end effector is wrist joint. The maximum number of iterations is set to 50, and the sampling number is set to 10, and convergent precisions are respectively set to

https://doi.org/10.1017/S0263574716000205 Published online by Cambridge University Press



(a) (b)

Fig. 7. An example for the obstacle-avoidance grasping of a virtual right arm. (a) Initial posture. (b) Grasping posture.

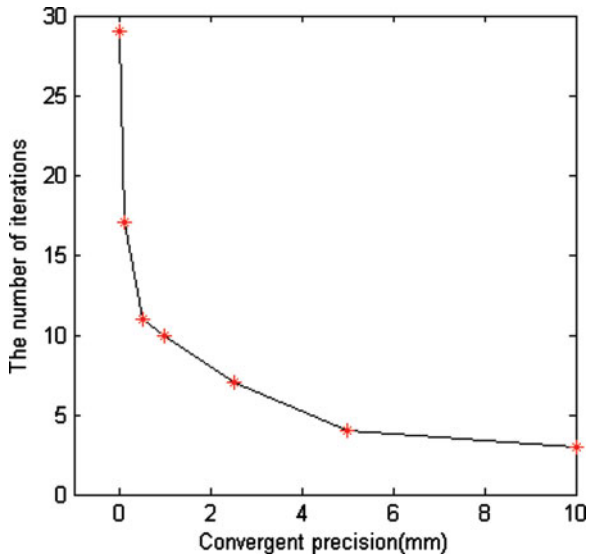


Fig. 8. The curve for the number of iterations changing with convergent precision.

$\boldsymbol{\varepsilon}$ = 10 mm, 5 mm, 2.5 mm, 1 mm, 0.5 mm, 0.1 mm and 0.01 mm. Besides, four 150 mm × 150 mm × 150 mm cubes are used to represent the obstacles while the coordinate frame located in the middle and forward of four cubes is used to describe the target position (see Fig. 7). The interference-checking is whether the minimum distance between the right arm and obstacles is greater than zero. For each convergent precision $\boldsymbol{\epsilon}$, the program is run 10 times. The curve for the number of iterations changing with convergent precision is listed in Fig. 8, and the plot for running time changing with convergent precision is presented in Fig. 9, and the plot for the distance between the wrist joint and target position changing with convergent precision is presented in Fig. 10. As shown in Fig. 8, in cases where loop iterations are less than 15 can ensure the convergence of the algorithm for average convergence precision ($\varepsilon \geq 0.5$ mm). However, the number of iterations obviously increases for high convergence precision (such as $\varepsilon = 0.1$ or $\varepsilon = 0.01$). As can be seen from Fig. 9, the increase of running time is usually no more than 2s with the increase of convergence precision. From Fig. 10, the distance between the wrist joint and the target position is dropped to the final value 10 mm from the original value 599.4 mm after four iterations. The experimental results show that the proposed algorithm has good convergence.

https://doi.org/10.1017/S0263574716000205 Published online by Cambridge University Press

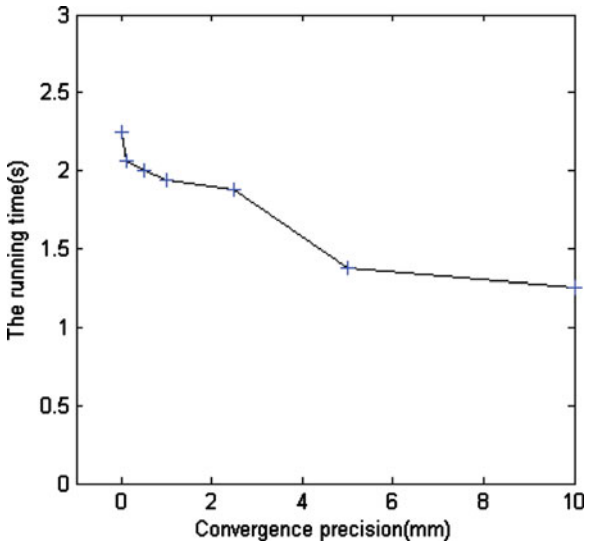


Fig. 9. The plot for running time changing with convergent precision.

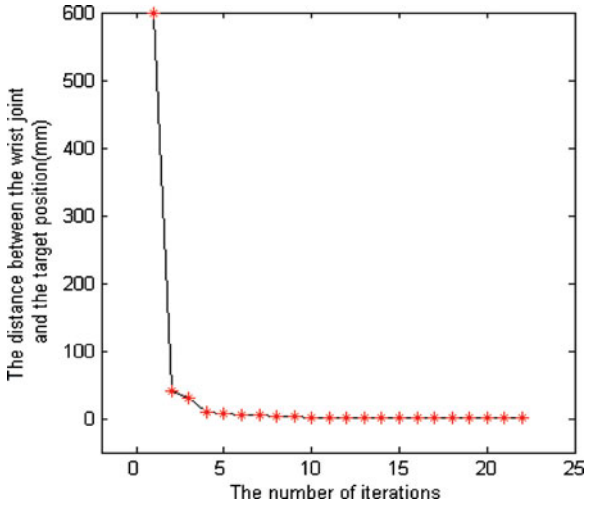


Fig. 10. The plot for the distance between the wrist joint and target position changing with convergent precision.

4. Motion Planning of Virtual Based on the Bi-RRT Algorithm

4.1. Problem definition

In Section 3.3, we know a virtual arm has seven joint angles, which can constitute a seven-dimensional configuration space $C(C \subseteq R^7)$. Let $q = \{\alpha_1, \alpha_2, \alpha_3, \alpha_4, \alpha_5, \alpha_6, \alpha_7\}$ be a configuration in C. If a joint angle $\alpha_k (k = 1 \sim 7)$ in a configuration q satisfies the constraint $\alpha_{k\ \min} \leq \alpha_k \leq \alpha_{k\ \max}$($\alpha_{k\ \min}$ and $\alpha_{k\ \max}$ are respectively the minimum and maximum values of α_k), we think the configuration q meets the condition of constraints, the flag $CS(q) =$ True. Then, all constrained configurations can constitute a constrained configuration space $C_c = \{q | q \in C, CS(q) = \text{True}\}$. Therefore, the motion planning of a virtual arm is the problem of finding collision-free motion between the initial configuration q_i and the target configuration q_t in a seven-dimensional configuration space C_c.

4.2. Bi-RRT algorithm

The detailed procedures of the Bi-RRT algorithm are presented as follows:

Input: The initial configuration q_i and the target configuration q_t; the maximum number N of nodes in a search tree; convergent precision ε.

https://doi.org/10.1017/S0263574716000205 Published online by Cambridge University Press

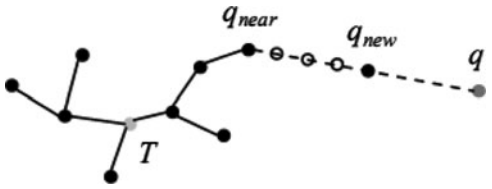


Fig. 11. The method for extending new nodes.

Output: The collision-free path between the initial configuration q_i and the target configuration q_t

1. T_a.Init(q_i), T_b.Init(q_t);
2. **While** T_a.Size()+T_b.Size()< N **do**
3. q_r = RandomState(C_c);
4. **If not** (Connect(q_r, T_a) = *Trapped*) **then**
5. **If** (Connect(q_{new}, T_b) = *Reached*)
6. **Return** Path(T_a, T_b);
7. Swap(T_a, T_b);
8. **Return** *Failure*;

Connect(q, T)

9. q_{near} = NearestNeighbor(T);
10. **If** Extend(q, q_{near}, q_{new})
11. T.AddNode(q_{new});
12. **If** $\rho(q, q_{new}) < \epsilon$
13. **Return** *Connected*;
14. **Else**
15. **Return** *Advanced*;
16. **Return** *Trapped*.

In above procedures, the search trees T_a and T_b are initialized first. The initial configuration q_i and the target configuration q_t are root nodes, and two trees T_a and T_b are constructed (see line 1). Then, a configuration q_r is randomly generated in a constrained configuration space C_c (see line 3). The tree T_a is extended to approach the node q_r (see line 4). If the extension is successful (Connect (q_r, T_a) doesn't return *Trapped*), T_a will continue to extend and an updated node q_{new} is added in T_a. In the next step, the tree T_b is extended to approach the node q_{new} (see line 5). If the function Connect (q_{new}, T_b) returns *Reached*, the tree T_b and the node q_{new} are connected. Thus, the trees T_a and T_b are connected, and the motion is found. Otherwise, swap T_a and T_b, and the search process is continued. If two trees T_a and T_b are connected after finite times growing, the program returns the found motion (see line 6). If the node number of two trees T_a and T_b adds up to more than the maximum number N and two trees T_a and T_b are unconnected, the program returns *Failure* (see line 8, no feasible path has been found).

The function Connect (q, T) is used to generate the node q_{new} from the tree T to the target node q (see line 9∼16), and returns the connection status information. The nearest neighbor node q_{near} has the minimum distance with the target node q in a tree T (see line 9). The function Extend (q, q_{near}, q_{new}) generates the node q_{new} when the node q_{near} is extended to approach the node q (see line 10). As the typical RRT algorithm only has a step size for generating updated node, the algorithm is inefficient. Then, the greed strategy and multiple extensions are adopted to find the node q_{new} from the node q_{near} to the node q, until the node q_{new} is closest to the node q or the node q_{new} has the interference (see Fig. 11). If the extension is failed, the connection function returns *Trapped*. Otherwise, the extension is successful, and q_{new} is added in T_a. In line 12, $\rho(q, q_{new})$ represents the Euclidean distance between the nodes q and q_{new}. If $\rho(q, q_{new}) < \epsilon$, the tree T and the node q are connected, return *Connected*; otherwise, return *Advanced*.

For the purpose of comparison, the experimental results of the RRT algorithm and Bi-RRT algorithm for two-dimensional planar motion planning are given in Fig. 12. Blue circles represent obstacles, red points are initial point and yellow points are target points, and red circles represent found paths. In Fig. 12(a), green lines represent search trees. The RRT algorithm only has a search

https://doi.org/10.1017/S0263574716000205 Published online by Cambridge University Press

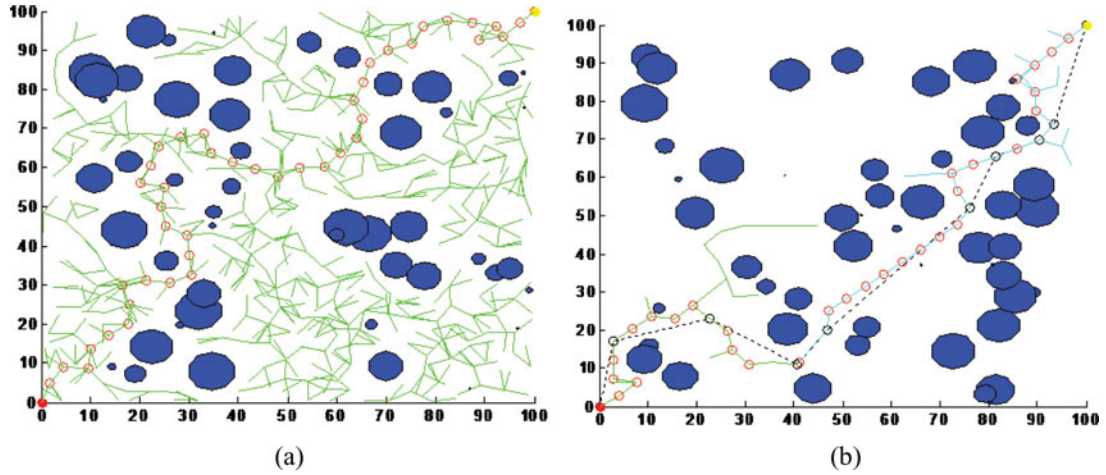


Fig. 12. The experimental results of comparing the RRT algorithm and Bi-RRT algorithm. (a) RRT algorithm. (b) Bi-RRT algorithm.

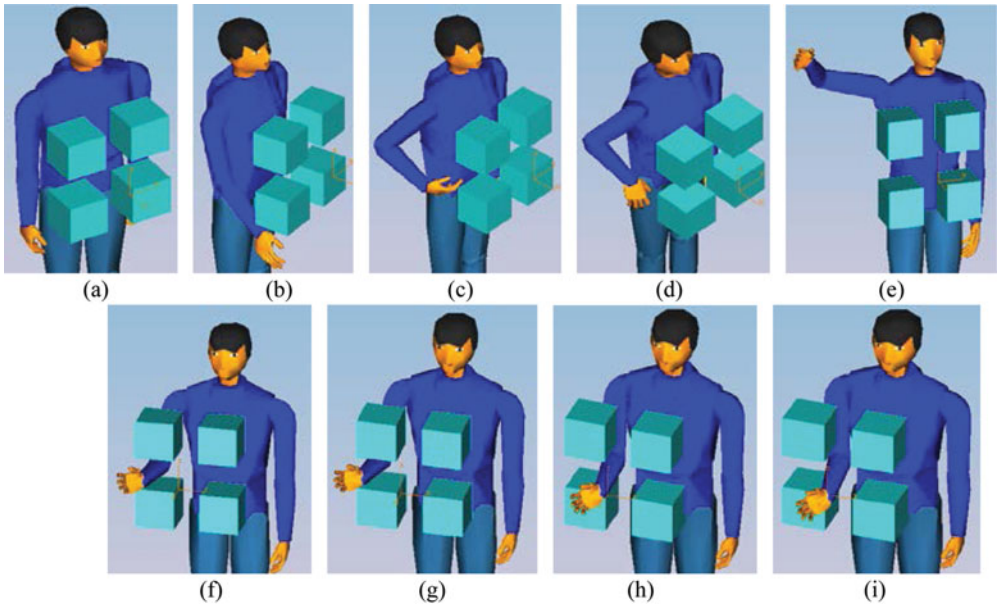


Fig. 13. The motion planning of a virtual arm based on the Bi-RRT algorithm.

tree whose root node is the initial point. In Fig. 12(b), the cyan line represents another search tree whose root node is the target point. The Bi-RRT algorithm has two search trees which begin to grow in opposite directions. Thus, the Bi-RRT algorithm has significant improvement in efficiency.

Different from the existing method, this paper directly uses the obstacle-avoidance FABRIK algorithm and Bi-RRT algorithm to explore arm motion planning in a seven-dimensional angle space automatically. In the following, the experiments for arm motion planning based on the Bi-RRT algorithm are performed on the second developing tool. The experimental subject and collision detection method are the same as the tests in Section 3.3. The initial posture and the grasping posture are respectively given in Fig. 13(a) and (i), and the grasping posture is obtained by using the obstacle-avoidance FABRIK algorithm in Section 3.3. Here, the Bi-RRT algorithm is introduced to explore the process postures according to the initial posture and grasping posture. The maximum size N of nodes is set to 100 in a search tree; the step size for extending search tree is a unit; and convergent precision ϵ is set to 5. The experimental results are listed in Fig. 13 and the angles of each joint are given in Table II. The running time of the experiment is 15.72 s; and the node number of two trees adds up to 33; and the simplified motions have nine nodes.

https://doi.org/10.1017/S0263574716000205 Published online by Cambridge University Press

Table II. The angles of each joint based on the Bi-RRT algorithm.

		The rotation angle of an arm joint (°)							
		Shoulder joint			Elbow joint	Wrist joint			Experimental
Type	Node	θ_1	θ_2	θ_3	θ_4	θ_5	θ_6	θ_7	results
Before being simplified	1	0	0	0	0	0	0	0	Fig. 14(a)
	2	12.45	−1.38	14.10	38.34	−6.42	−17.73	−20.01	Fig. 14(b)
	3	−39.37	9.90	1.39	105.09	−12.90	37.948	35.66	Fig. 14(c)
	4	−32.54	98.67	−23.79	114.27	17.86	−8.58	35.23	Fig. 14(d)
	5	44.26	106.04	−12.86	66.79	−9.25	−16.56	56.17	Fig. 14(e)
	6	53.11	−0.49	6.35	41.39	20.06	−38.00	14.05	Fig. 14(f)
	7	54.85	−0.05	6.93	42.98	20.79	−40.51	13.67	Fig. 14(g)
	8	55.44	−15.33	9.99	34.72	22.94	−44.22	19.95	Fig. 14(h)
	9	54.30	−18.53	12.16	32.15	25.00	−45.00	20.05	Fig. 14(i)
After being simplified	1	0	0	0	0	0	0	0	Fig. 14(a)
	2	44.26	106.04	−12.86	66.79	−9.25	−16.56	56.17	Fig. 14(e)
	3	53.11	−0.49	6.35	41.39	20.06	−38.00	14.05	Fig. 14(f)
	4	55.44	−15.33	9.99	34.72	22.94	−44.22	19.95	Fig. 14(h)
	5	54.30	−18.53	12.16	32.15	25.00	−45.00	20.05	Fig. 14(i)

4.3. The post-processing technique for the Bi-RRT algorithm

As the Bi-RRT algorithm is a random algorithm, the planned motions usually have some redundant process postures. To remove redundant nodes, we need to simplify the planned motion. The criterion is whether a virtual arm has interference from a node to the next node in a simplified motion. Therefore, we need to get process postures of a virtual arm. As each node is the combination of seven joint angles, the linear interpolation between an initial angle and terminal angle is adopted to get rotation angles of process postures. As the initial angle and terminal angle satisfy the constraints, and the constrained domains are continuous and no-concave, the interpolation points will be within the constrained boundary. If the distance between the interpolations is oversized, some of collision detections might be missed. To resolve this problem, the number of interpolations should be increased appropriately. The experiments show that 10∼15 interpolations is proper for the problem in Fig. 13.

The post-processing technique for the Bi-RRT algorithm is given in Fig. 14. First, the node p_1 in the original motion set P ($P = p_1, p_2, \ldots$.) is put into the simplified motion set S, set $s_1 = p_1$. Second, let s_1 be the starting point, connect the nodes s_1 and p_3; let the connecting line be interpolated, some process postures are generated. And check whether each posture has the interference. If the interference doesn't exist, the node p_2 is redundant and should be removed. Similar to the node p_3, the nodes p_4, p_5... are implemented the same procedures, until an interpolation has the interference. If the interpolation between the nodes s_1 and p_i has the interference, the node p_i is called as an intervention node; and the previous node of p_i is put into S, set $s_2 = p_{i-1}$ (see p_i in Fig. 14). Then, let s_2 be the starting point, the nodes p_{i+1}, p_{i+2}... are implemented interpolation calculations and interference detections, the previous node of the intervention node is put into S. Finally, the remainder nodes in the original motion set P are implemented the same procedures. In Table II, nine nodes are simplified to five nodes, four redundant motions are removed.

5. Experimental Results

A prototype system for collision-free motion planning of a virtual arm is developed based on Tecnomatix/Process Simulate and VS2005. Meanwhile, the program packages of the obstacle-avoidance FABRIK algorithm and Bi-RRT motion planning for a virtual arm are built based on the secondary development technology. The frame diagram of the prototype system is given in Fig. 15. First, collision-free grasping posture of a virtual arm is obtained by using the program package of the obstacle-avoidance FABRIK algorithm according to the target position. Then, some collision-free

https://doi.org/10.1017/S0263574716000205 Published online by Cambridge University Press

Table III. Implement parameters of two examples for the motion planning of a virtual arm.

	Running time(s)			The number of nodes	
	Obstacle-avoidance FABRIK algorithm	Bi-RRT motion planning	The number of iterations	Before being simplified	After being simplified
Example #1	0.532	54.6	12	16	7
Example #2	0.359	25.6	4	12	8

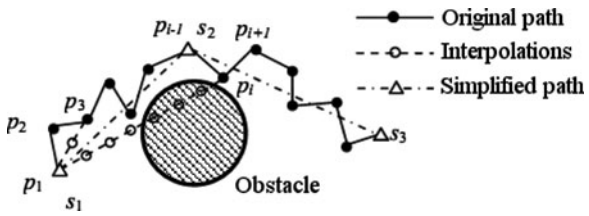


Fig. 14. The post-processing technique for the Bi-RRT algorithm.

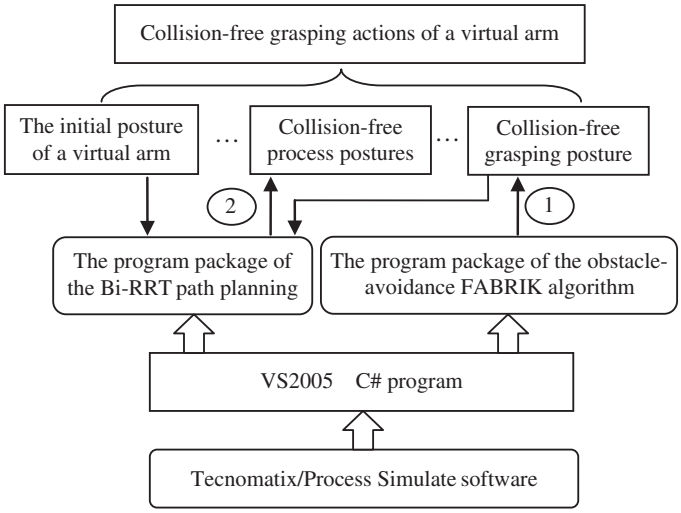


Fig. 15. The frame diagram of the prototype system.

process postures are gained by using the program package of the Bi-RRT motion planning based on the initial posture and the grasping posture. Finally, the combination of the initial posture, collision-free process postures and the grasping posture can achieve collision-free actions of a virtual arm.

The flow diagram for obstacle-avoidance FABRIK algorithm is listed in Fig. 16. If $\sum_{j=1}^{n-1} d_j < |P_1 - t|$, the target position is reachable in line; otherwise, the target position is unreachable in line. And if the minimum distance between the interpolation point of a connecting rod and the center of a spherical bounding box is larger than the radius of the bounding box, the interference exists; otherwise, the interference does not exist (see section 3.2). Let $\alpha_k (k = 1, 2, ..., 7)$ be the bound of rotation angle, if $\alpha_k > \alpha_{k\,\max}$, set $\alpha_k = \alpha_{k\,\max}$; if $\alpha_k < \alpha_{k\,\min}$, set $\alpha_k = \alpha_{k\,\min}$.

In the following, two experiments are conducted for the proposed method. These tests were performed on a computer with an Intel 3.0 GHz CPU and 2.0 GB RAM. For obstacle-avoidance FABRIK algorithm, the maximum number of iterations, the sampling number and convergent precision ϵ_1 are respectively set to 50, 15 and 1.0 mm. For Bi-RRT motion planning, the maximum size of nodes is set to 100 in a search tree; the step size for extending the search tree is a unit; and convergent precision ϵ_2 is set to 5°. The experimental results are presented in Table III and Fig. 17. In example #1, a virtual right arm has successfully passed through the square hole of a partition to grasp a pink cube. The running time of the FABRIK algorithm is 0.532 s and the number of iterations is 12 while the running time of the Bi-RRT algorithm is 56.4 s, and the number of nodes before

https://doi.org/10.1017/S0263574716000205 Published online by Cambridge University Press

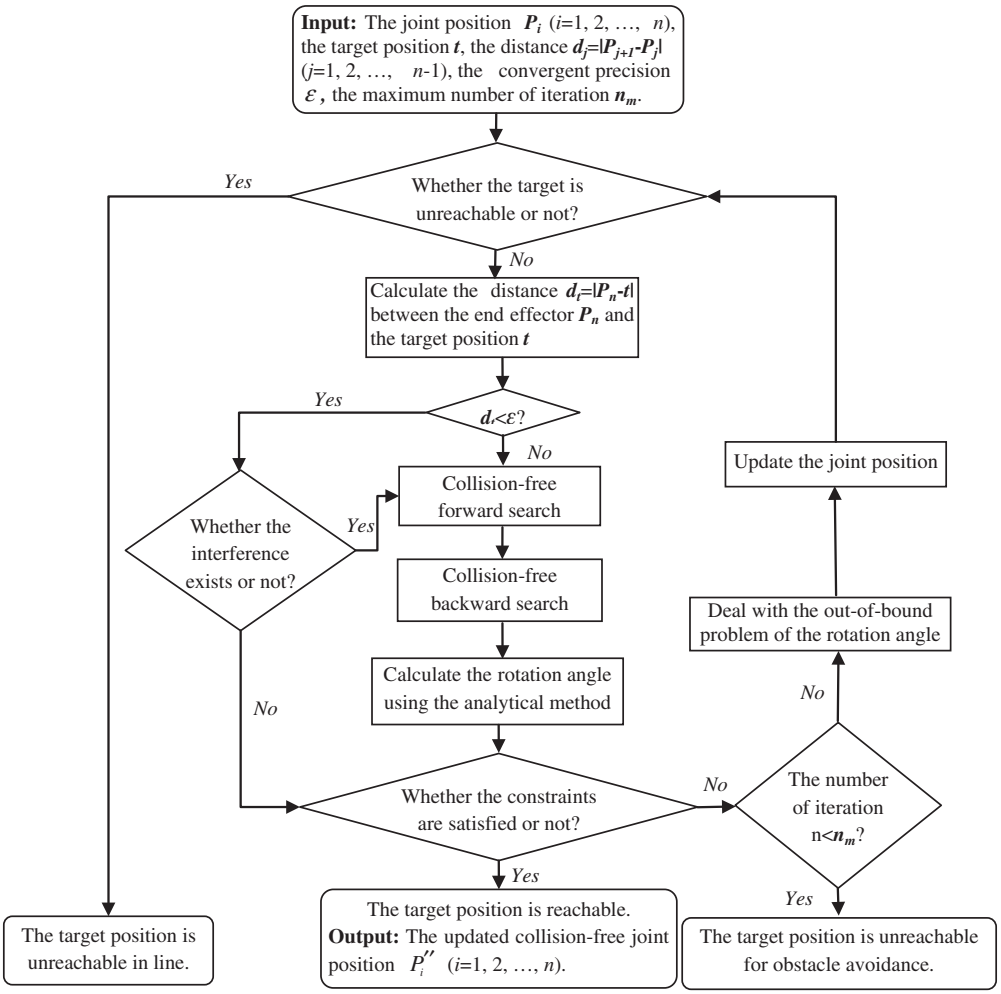


Fig. 16. The flow diagram for the obstacle-avoidance FABRIK algorithm.

being simplified is 16. Using the post-processing technique, 16 nodes are simplified to seven nodes in the planned motion. In example #2, a virtual right arm has successfully avoided the boss and partition to grasp the pink cube on the shelf. The running time of the FABRIK algorithm is 0.359 s and the number of iterations is four while the running time of the Bi-RRT algorithm is 25.6 s, and the number of nodes before being simplified is 12. Using the post-processing technique, 12 nodes are simplified to eight nodes in the planned motion. The experimental results show the proposed method for collision-free motion planning of a virtual arm is feasible.

6. Conclusions

For virtual maintenance and assembly operations, most of the actions are performed by the movements of upper limbs. As the work environment is complex, a virtual arm is easy to interfere with its surrounding scenarios. Collision-free motion planning of a virtual arm is an intractable task in high-interference environments. Therefore, an approach for collision-free motion planning of a virtual arm based on the FABRIK algorithm is proposed in this paper. First, a random rotation strategy and local optimum-seeking technology are introduced to improve the FABRIK algorithm. The improvement FABRIK algorithm has resolved the problems that the FABRIK algorithm is unable to add joint constraints and avoid obstacles. Moreover, the improvement FABRIK algorithm has good convergence and low computational cost. Then, the Bi-RRT algorithm is adopted to explore the process postures from a given initial posture to the final grasping posture. Different from the existing method, the proposed Bi-RRT algorithm in this paper plans the motions of a virtual arm in a seven-dimensional

https://doi.org/10.1017/S0263574716000205 Published online by Cambridge University Press

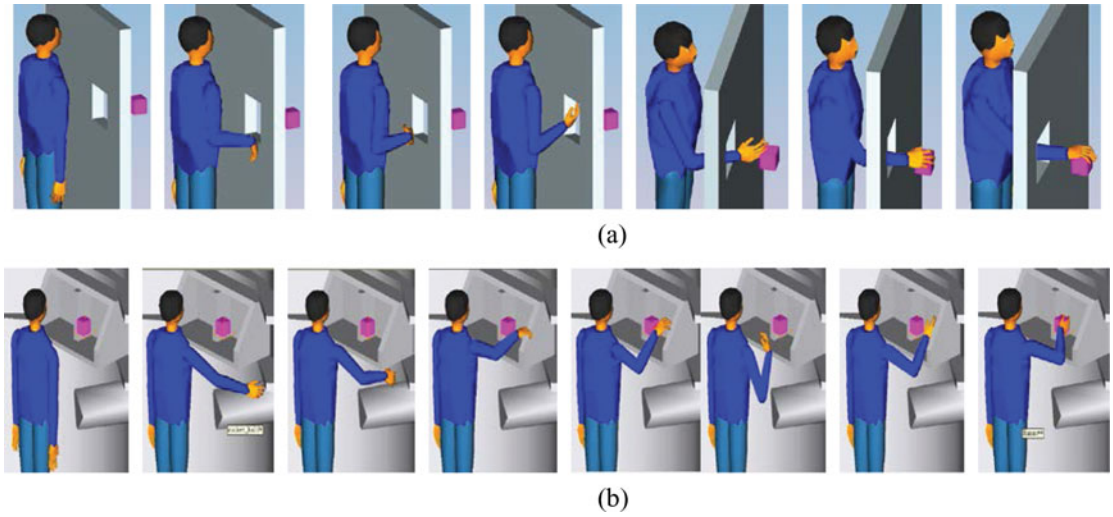


Fig. 17. Two examples for the motion planning of a virtual arm. (a) Example #1: A virtual right arm passed through the square hole of a partition to grasp a pink cube. (b) Example #2: A virtual right arm avoided the boss and partition to grasp a pink cube on the shelf.

angle space, and the final grasping posture is automatically designed using the improvement FABRIK algorithm instead of the manual design. Finally, the post-processing technique is introduced to remove redundant nodes from the planned motions. The procedure has resolved the problem that the Bi-RRT algorithm is a random algorithm. The experimental results show the proposed method for collision-free motion planning of a virtual arm is feasible.

However, the proposed approach still suffers from some shortcomings. First, as a random rotation strategy is introduced to avoid obstacles, each experimental result may be different. Second, the result of the Bi-RRT algorithm is a feasible solution instead of the optimal one. In the future, we will improve the approach for avoiding obstacles, enhance the stability of the algorithm and optimize the results of the obstacle-avoidance FABRIK algorithm and Bi-RRT algorithm.

References

1. A. Aristidou and J. Lasenby, "FABRIK: A fast, iterative solver for the inverse kinematics problem," *Graph. Models* **73**(5), 243–260 (2011).
2. A. Aristidou, Y. Chrysanthou and J. Lasenby "Extending FABRIK with model constraints," *Comput. Animat. Virtual Worlds* DOI: 10.1002/cav.1630. (2015).
3. S. M. LaValle, "Rapidly-exploring random trees: A new tool for path planning," *Technical Report* (Computer Science Department, Iowa State University, 1998) (TR 98-11).
4. J. J. Kuffner and S. M. LaValle, "RRT-connect: An efficient approach to single-query path planning," *IEEE Int. Conf. Robot. Autom.* **2**, 995–1001 (2000).
5. G. Schreiber and G. Hirzinger, "Singularity Consistent Inverse Kinematics by Enhancing the Jacobian Transpose," **In**: *Advances in Robot Kinematics: Analysis and Control* (L. Jadran and M. L. Husty, eds.) (Springer, Netherlands, 1998), pp. 475–482.
6. Z. Bhatti, A. Shah, F. Shahidi and M. Karbisi, "Forward and inverse kinematics seamless matching using Jacobian," *Sindh Univ. Res. J. (SURJ)* **45**(2), 387–392 (2014).
7. Y. Nakamura and H. Hanafusa, "Inverse kinematic solutions with singularity robustness for robot manipulator control," *J. Dyn. Syst. Meas. Control*, **108**(3), 163–171 (1986).
8. L. Cui and J. S. Dai, "Reciprocity-based singular value decomposition for inverse kinematic analysis of the metamorphic multifingered hand," *J. Mech. Robot.* **4**(3), 034502 (2012).
9. C. W. Wampler, "Manipulator inverse kinematic solutions based on vector formulations and damped least-squares methods," *IEEE Trans. Syst. Man Cybern.* **16**(1), 93–101 (1986).
10. L. M. Phuoc, P. Martinet, S. Lee and H. Kim, "Damped least square based genetic algorithm with Ggaussian distribution of damping factor for singularity-robust inverse kinematics," *J. Mech. Sci. Technol.* **22**(7), 1330–1338 (2008).
11. C. Welman, "*Inverse Kinematics And Geometric Constraints For Articulated Figure Manipulation*," Master Dissertation, (Simon Fraser University, Department of Computer Science, 1993).

https://doi.org/10.1017/S0263574716000205 Published online by Cambridge University Press

12. M. Meredith and S. Maddock, "Adapting motion capture data using weighted real-time inverse kinematics," *Comput. Entertainment (CIE)* **3**(1), 1–15 (2005).
13. S. Tejomurtula and S. Kak, "Inverse kinematics in robotics using neural networks," *Inf. Sci.* **116**(2), 147–164 (1999).
14. B. Karlık and S. Aydın, "An improved approach to the solution of inverse kinematics problems for robot manipulators," *Eng. Appl. Artif. Intell.* **13**(2), 159–164 (2000).
15. R. V. Mayorga and P. Sanongboon, "Inverse kinematics and geometrically bounded singularities prevention of redundant manipulators: An artificial neural network approach," *Robot. Auton. Syst.* **53**(3), 164–176 (2005).
16. R. Köker, "A genetic algorithm approach to a neural-network-based inverse kinematics solution of robotic manipulators based on error minimization," *Inf. Sci.* **222**(10), 528–543 (2013).
17. R. Köker, T. Çakar and Y. Sari, "A neural-network committee machine approach to the inverse kinematics problem solution of robotic manipulators," *Eng. Comput.* **30**(4), 641–649 (2014).
18. B. Daya, S. Khawandi and M. Akoum, "Applying neural network architecture for inverse kinematics problem in robotics," *J. Softw. Eng. Appl.* **3**(03), 230 (2010).
19. L. C. Wang and C. C. Chen, "A combined optimization method for solving the inverse kinematics problems of mechanical manipulators," *IEEE Trans. Robot. Autom.* **7**(4), 489–499 (1991).
20. A. A. Canutescu and R. L. Dunbrack, "Cyclic coordinate descent: A robotics algorithm for protein loop closure," *Protein Sci.* **12**(5), 963–972 (2003).
21. A. L. Olsen and H. G. Petersen, "Inverse kinematics by numerical and analytical cyclic coordinate descent," *Robotica* **29**(04), 619–626 (2011).
22. B. Kenwright, "Inverse kinematics–cyclic coordinate descent (CCD)," *J. Graph. Tools* **16**(4), 177–217 (2012).
23. R. Müller-Cajar and R. Mukundan, "Triangulation: A New Algorithm for Inverse Kinematics," *Proceedings of the Image and Vision Computing*, University of Waikato Hamilton, New Zealand, (2007) pp. 181–186.
24. R. Mukundan, "A robust inverse kinematics algorithm for animating a joint chain," *Int. J. Comput. Appl. Technol.* **34**(4), 303–308 (2009).
25. O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," *The Int. J. Robot. Res.* **5**(1), 90–98 (1986).
26. Y. Koren and J. Borenstein, "Potential Field Methods and Their Inherent Limitations for Mobile Robot Navigation," *IEEE International Conference on Robotics and Automation*, Sacramento, California, (1991) pp. 1398–1404.
27. J. Ren, K. McIsaac and R. V. Patel, "Modified Newton's method applied to potential field-based navigation for mobile robots," *IEEE Trans. Robot.* **22**(2), 384–391 (2006).
28. B. Xiao, L. Yu, S. Li and R. Chen, "Research of escaping local minima strategy for artificial potential field," *J. Syst. Simul.* **19**(19), 4495–4503 (2007).
29. Q. Jia and X. Wang, "Path planning for mobile robots based on a modified potential model," *IEEE, International Conference on Mechatronics and Automation*, Changchun, China, (2009) pp. 4946–4951.
30. J. Lee, Y. Nam, S. Hong and W. Cho "New potential functions with random force algorithms using potential field method," *J. Intell. Robot. Syst.* **66**(3), 303–319 (2012).
31. J. Velagic, B. Lacevic and N. Osmic, "Efficient Path Planning Algorithm for Mobile Robot Navigation with a Local Minima Problem Solving," *IEEE International Conference on Industrial Technology*, Mumbai, India, (2006) pp. 2325–2330.
32. R. Raja, A. Dutta and K. S. Venkatesh, "New potential field method for rough terrain path planning using genetic algorithm for a 6-wheel rover," *Robot. Auton. Syst.* **72**(10), 295–306 (2015).
33. L. E. Kavraki, P. Švestka, J. C. Latombe and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Trans. Robot. Autom.* **12**(4), 566–580 (1996).
34. L. E. Kavraki, M. N. Kolountzakis and J. C. Latombe, "Analysis of probabilistic roadmaps for path planning," *IEEE Trans. Robot. Autom.* **14**(1), 166–171 (1998).
35. J. J. Kuffner and S. M. LaValle, "RRT-connect: An Efficient Approach to Single-Query Path Planning," *IEEE International Conference on Robotics and Automation*, San Francisco, California, **2**, 995–1001 (2000).
36. S. M. LaValle and J. J. Kuffner, "Randomized kinodynamic planning," *The Int. J. Robot. Res.* **20**(5), 378–400 (2001).
37. D. Hsu, J. C. Latombe and H. Kurniawati, "On the probabilistic foundations of probabilistic roadmap planning," *The Int. J. Robot. Res.* **25**(7), 627–643 (2006).
38. M.T. Rantanen and M. Juhola, "A configuration deactivation algorithm for boosting probabilistic roadmap planning of robots," *Int. J. Autom. Comp.* **9**(2), 155–164 (2012).
39. M. Kothari and I. Postlethwaite, "A probabilistically robust path planning algorithm for UAVs using rapidly-exploring random trees," *J. Intell. Robot. Syst.* **71**(2), 231–253 (2013).
40. S. M. LaValle and J. J. Kuffner, "Randomized kinodynamic planning," *The Int. J. Robot. Res.* **20**(5), 378–400 (2011).
41. A. H. Qureshi and Y. Ayaz, "Intelligent bidirectional rapidly-exploring random trees for optimal motion planning in complex cluttered environments," *Robot. Auton. Syst.* **68**(6), 1–11 (2015).

https://doi.org/10.1017/S0263574716000205 Published online by Cambridge University Press