Special Issue Article

Advances in Mechanical Engineering

Advances in Mechanical Engineering
2015, Vol. 7(12) 1–9
© The Author(s) 2015
DOI: 10.1177/1687814015619768
aime.sagepub.com
SAGE

Autonomous path planning solution for industrial robot manipulator using backpropagation algorithm

PeiJiang Yuan, Feng Su, ZhenYun Shi, TianMiao Wang and DongDong Chen

Abstract
Here, we propose an autonomous path planning solution using backpropagation algorithm. The mechanism of movement used by humans in controlling their arms is analyzed and then applied to control a robot manipulator. Autonomous path planning solution is a numerical method. The model of industrial robot manipulator used in this article is a KUKA KR 210 R2700 EXTRA robot. In order to show the performance of the autonomous path planning solution, an experiment validation of path tracking is provided. Experiment validation consists of implementation of the autonomous path planning solution and the control of physical robot. The process of converging to target solution is provided. The mean absolute error of position for tool center point is also analyzed. Comparison between autonomous path planning solution and the numerical methods based on Newton–Raphson algorithm is provided to demonstrate the efficiency and accuracy of the autonomous path planning solution.

Keywords
Industrial engineering, robotics, kinematics, inverse analysis, numerical analysis

Date received: 19 July 2015; accepted: 5 November 2015

Academic Editor: Zhijun Li

Introduction

The control and movement of arm play an important role in our daily life. It is very easy for humans to reach visual targets, such as pressing several keys. In many behavioral experiments, subjects were asked to move their arm till their fingertip reaches a dot in space which was set at a random position.[1] Even we can control our hand to reach proprioceptive targets in the absence of vision.[2] In addition, humans do not know the accurate model of their body, such as the length of their arm and fingers, and they also do not need the accurate model of the world, such as the Cartesian coordinate of the original position and the endpoint. However, a precise model and a good algorithm are the basics of controlling an industrial robot. Current industrial solution is not intelligent enough. We can make robot more intelligent by mimicking the mode of human motion.

The inverse kinematics (IK) problem is a major theme in most researches and applications of industrial robot manipulators, such as path planning, offline programming, arc welding, and painting and machining. While the manipulator is a complex system, an ideal solution of the IK problem is always difficult to solve. In the literature, most of the methods to solve the IK problem can be classified into analytical methods and numerical methods.[3] The analytical methods mainly consist of algebraic methods[4,5] and geometric methods.[6] However, some IK problems cannot be solved if the robot manipulators do not have closed form.

Beihang University, Beijing, China

Corresponding author:
ZhenYun Shi, Beihang University, Beijing 100191, China.
Email: shichong1983623@hotmail.com

The numerical methods to solve IK problem can also be classified into two types.[3] The first type of numerical methods consists of solutions based on Newton–Raphson (NR) method,[7] and the solutions are based on predictor–corrector algorithms.[8] The singularity problem is always difficult to solve when this kind of method is used, although some solutions are proposed to avoid the singularities.[9] The other types of numerical methods use different gradient-based nonlinear algorithms to solve IK problem, and this kind of method usually does not use Jacobian matrix.[10] However, the accuracy of numerical methods is always influenced by the initial value of joint variables.

Some intelligent methods for solving IK problem based on intelligent algorithm also have been developed, such as approaches based on artificial neural network (ANN),[11–13] fuzzy system,[14] and genetic algorithm.[15] In addition, there are many methods based on other theory, such as conformal geometric space theory,[16] double quaternion,[17] and planar curves intersecting.[18] Backstepping method is also a powerful technique for the robotic system.[19] However, these methods have not yet been widely used.

Although all above-mentioned methods play an important role in the development of robot manipulator, no one is comparable to the method used by mankind to control their arm. By mimicking arm movement, we propose an autonomous path planning solution (APPS) for industrial robot manipulator using backpropagation (BP) algorithm, which has a good performance in solving IK problem. We analyzed the mechanism of human beings for controlling and moving their arms and then introduced it to control the industrial robot manipulator. Since the APPS is not involved with Jacobian matrix, it can converge to solution even when singularity exists.

The rest of this article is organized as follows. Section "Control and movement of arm in mankind" describes the mechanism of arm movement for mankind. Section "Autonomous path planning solution" introduces the mechanism of arm movement into the method which is used to control robot manipulator, and the APPS based on BP algorithm is explained. Section "Experiment validation" presents experiment validation to show the verification of the APPS. Section "Comparison between APPS and NR" presents comparison between APPS and the numerical methods based on NR algorithm. The conclusion is presented in section "Conclusion."

Control and movement of arm in mankind

Because of the complexity of the central nervous system (CNS), we still cannot explain how humans control their arm and how proprioception is formed from the microperspective.[1] But the fact is that no sophisticated industrial manipulator which has accurate model and global information is comparable to the arm of humans in the aspect of flexibility.

The process of reaching an endpoint for a person using his arm can be classified into three phases: in the first phase, the tester helds his or her arm still in an initial condition; the second phase was an intermediate state that the tester was moving his or her arm, but the finger did not reach the endpoint; in the third phase, the tester's finger reached the endpoint set in the space (Figure 1). The whole process was under the control of CNS, no matter the endpoint was visible or remembered (Figure 2). Using this method, CNS just needs to determine whether fingers reached the endpoint and what to do next, without performing complex calculations. In the following sections, we will introduce this

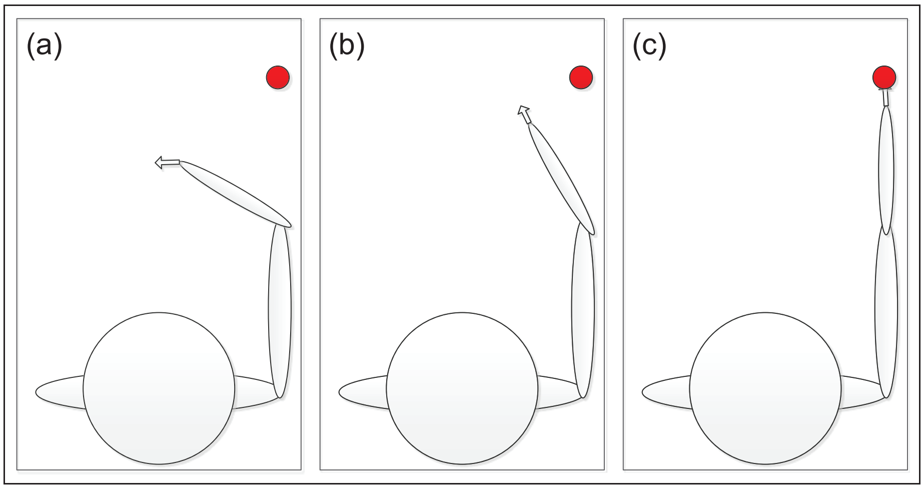

Figure 1. Process of reaching an endpoint: (a) initial condition, (b) intermediate condition, and (c) final condition.

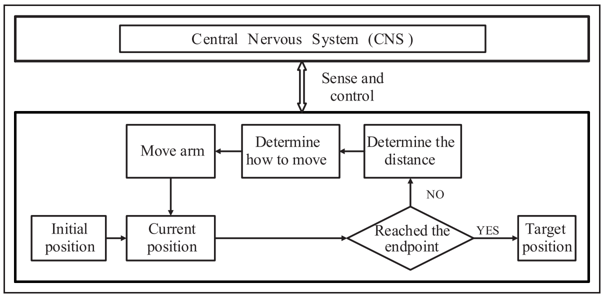

Figure 2. Mechanism of arm movement.

mechanism into the control of industrial robot manipulator.

APPS

APPS for industrial robot manipulator using BP algorithm is proposed according to the mechanism of arm movement.

Forward kinematics

In the following discussion, we use the model of KUKA KR robot which is a 6R industrial robot. We build the homogeneous transformation matrix for a single joint

$$ {}^{i-1}_{i}T(\theta_i) = \begin{bmatrix} c\theta_i & -s\theta_i & 0 & a_{i-1} \\ s\theta_i c\alpha_{i-1} & c\theta_i c\alpha_{i-1} & -s\alpha_{i-1} & -s\alpha_{i-1}d_i \\ s\theta_i s\alpha_{i-1} & c\theta_i s\alpha_{i-1} & c\alpha_{i-1} & c\alpha_{i-1}d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{1}$$

where s and c denote the sin and cos functions. We can get the forward kinematics of the tool center point (TCP) of KUKA KR robot with respect to the base frame through the following equation

$$ {}^{0}_{6}T = {}^{0}_{1}T(\theta_1){}^{1}_{2}T(\theta_2){}^{2}_{3}T(\theta_3){}^{3}_{4}T(\theta_4){}^{4}_{5}T(\theta_5){}^{5}_{6}T(\theta_6) \tag{2}$$

where $\theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6$ are the joint variables. ${}^{0}_{6}T$ depicts the position and orientation of the TCP of KUKA KR robot. We can get an alternative representation of equation (2) as follows

$$ {}^{0}_{6}T = \begin{bmatrix} r_{11} & r_{12} & r_{13} & r_{14} \\ r_{21} & r_{22} & r_{23} & r_{24} \\ r_{31} & r_{32} & r_{33} & r_{34} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3}$$

Error between the current condition and the target condition

We can set a target condition for the TCP by determining the value of each joint, choosing $\theta_{d1}, \theta_{d2}, \theta_{d3}, \theta_{d4}, \theta_{d5}, \theta_{d6}$ as the desired value of the joint variables $(\theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6)$. Using equations (2) and (3), we get the goal matrix of ${}^{0}_{6}T_d$

$$ {}^{0}_{6}T_d = {}^{0}_{1}T(\theta_{d1}){}^{1}_{2}T(\theta_{d2}){}^{2}_{3}T(\theta_{d3}){}^{3}_{4}T(\theta_{d4}){}^{4}_{5}T(\theta_{d5}){}^{5}_{6}T(\theta_{d6}) \tag{4}$$

$$ {}^{0}_{6}T_d = \begin{bmatrix} r_{d11} & r_{d12} & r_{d13} & r_{d14} \\ r_{d21} & r_{d22} & r_{d23} & r_{d24} \\ r_{d31} & r_{d32} & r_{d33} & r_{d34} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5}$$

We can also set a target condition for the TCP by determining the elements of ${}^{0}_{6}T_d$ (5) directly. Then we get the current value of joint variables $(\theta_{c1}, \theta_{c2}, \theta_{c3}, \theta_{d4}, \theta_{c5}, \theta_{c6})$. The current matrix of ${}^{0}_{6}T_c$ is determined using equations (2) and (3)

$$ {}^{0}_{6}T_c = {}^{0}_{1}T(\theta_{c1}){}^{1}_{2}T(\theta_{c2}){}^{2}_{3}T(\theta_{c3}){}^{3}_{4}T(\theta_{c4}){}^{4}_{5}T(\theta_{c5}){}^{5}_{6}T(\theta_{c6}) \tag{6}$$

$$ {}^{0}_{6}T_c = \begin{bmatrix} r_{c11} & r_{c12} & r_{c13} & r_{c14} \\ r_{c21} & r_{c22} & r_{c23} & r_{c24} \\ r_{c31} & r_{c32} & r_{c33} & r_{c34} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{7}$$

We define the error E using equations (5) and (7) as follows

$$ \Delta T = {}^{0}_{6}T_d - {}^{0}_{6}T_c \tag{8}$$

$$ \Delta T = \begin{bmatrix} r_{d11} - r_{c11} & r_{d12} - r_{c12} & r_{d13} - r_{c13} & r_{d14} - r_{c14} \\ r_{d21} - r_{c21} & r_{d22} - r_{c22} & r_{d23} - r_{c23} & r_{d24} - r_{c24} \\ r_{d31} - r_{c31} & r_{d32} - r_{c32} & r_{d33} - r_{c33} & r_{d34} - r_{c34} \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{9}$$

An alternative representation of ΔT can be written as

$$ \Delta T = \begin{bmatrix} e_{11} & e_{12} & e_{13} & e_{14} \\ e_{21} & e_{22} & e_{23} & e_{24} \\ e_{31} & e_{32} & e_{33} & e_{34} \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{10}$$

$$ E = \frac{1}{2}\sum_{i=1}^{3}\sum_{j=1}^{4} e_{ij}^2 \tag{11}$$

The value of error E depicts the distance between the current position and the target position. In order to judge whether the TCP has reached the destination, we set a constant value of E_0 which was used to compare with the current value of E. We can conclude that the TCP has reached the destination if $E \leq E_0$. Let $\theta'_{c1}, \theta'_{c2}, \theta'_{c3}, \theta'_{c4}, \theta'_{c5}, \theta'_{c6}$ as the result value of joint variables. The result value of each joint might have exceeded the range of the corresponding joint, so it is possible to check on the range of each joint and adjust the result value

$$\theta_{dk} = \begin{cases} \theta'_{ck} & \text{where} \quad \theta'_{ck} \in \left[R_k^{min} \quad R_k^{max}\right]; \\ \theta'_{ck} + 2\pi & \text{where} \quad \theta'_{ck} < R_k^{min}; \\ \theta'_{ck} - 2\pi & \text{where} \quad \theta'_{ck} > R_k^{max} \end{cases} \tag{12}$$

where $k = 1, 2, 3, 4, 5, 6$. θ_{dk} is the target angle of joint k. The range of joint k is $[R_k^{min} \quad R_k^{max}]$. It is obvious that $\sin\theta_{dk} = \sin\theta'_{ck}$, $\cos\theta_{dk} = \cos\theta'_{ck}$, so the position and orientation of TCP are not changed. We should determine how to control the robot to reduce error using BP algorithm if $E > E_0$.

BP

BP algorithm is widely used in ANN, and this type of ANN has been applied to many areas successfully, such as traffic noise prediction,[20,21] landslide susceptibility mapping,[22] and medical research.[23] If $E > E_0$, we can obtain the value of incremental joint angles $(d\theta_{c1}, d\theta_{c2}, d\theta_{c3}, d\theta_{d4}, d\theta_{c5}, d\theta_{c6})$ using BP algorithm

$$d\theta_{ck} = -\eta \frac{\partial E}{\partial \theta_{ck}} \tag{13}$$

where $k = 1, 2, 3, 4, 5, 6$ and η denotes learning rate. In this article, η is a very important parameter. The APPS would be unstable if the value of η is too big, and the APPS would be very time-consuming if the value of η is too small. In order to ensure good stability and rapidity, we performed lots of experiments to determine the best value of η as illustrated in the Supplementary Materials, where different values of learning speed ($\eta = 1$, 0.5, 0.1, 0.05) are discussed. We performed a trade-off study relating to the learning speed and the performance of APPS. Using equation (11), we can get the following equation

$$\frac{\partial E}{\partial \theta_{ck}} = \sum_{i=1}^{3}\sum_{j=1}^{4} e_{ij} \frac{\partial e_{ij}}{\partial \theta_{ck}} \tag{14}$$

Using equations (9) and (10), equation (14) can be changed as follows

$$\frac{\partial E}{\partial \theta_{ck}} = -\sum_{i=1}^{3}\sum_{j=1}^{4} \left(r_{dij} - r_{cij}\right) \frac{\partial r_{cij}}{\partial \theta_{ck}} \tag{15}$$

Using equation (7), we can get the following equation

$$\frac{\partial {}_6^0T_c}{\partial \theta_{ck}} = \begin{bmatrix} \frac{\partial r_{c11}}{\partial \theta_{ck}} & \frac{\partial r_{c12}}{\partial \theta_{ck}} & \frac{\partial r_{c13}}{\partial \theta_{ck}} & \frac{\partial r_{c14}}{\partial \theta_{ck}} \\ \frac{\partial r_{c21}}{\partial \theta_{ck}} & \frac{\partial r_{c22}}{\partial \theta_{ck}} & \frac{\partial r_{c23}}{\partial \theta_{ck}} & \frac{\partial r_{c24}}{\partial \theta_{ck}} \\ \frac{\partial r_{c31}}{\partial \theta_{ck}} & \frac{\partial r_{c32}}{\partial \theta_{ck}} & \frac{\partial r_{c33}}{\partial \theta_{ck}} & \frac{\partial r_{c34}}{\partial \theta_{ck}} \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{16}$$

We can get equations (17), (18), and (19), where $k = 2, 3, 4, 5$, by changing equation (16) using equation (6)

$$\frac{\partial {}_6^0T_c}{\partial \theta_{c1}} = \frac{\partial {}_1^0T(\theta_{c1})}{\partial \theta_{c1}} {}_2^1T(\theta_{c2}) {}_3^2T(\theta_{c3}) {}_4^3T(\theta_{c4}) {}_5^4T(\theta_{c5}) {}_6^5T(\theta_{c6}) \tag{17}$$

$$\frac{\partial {}_6^0T_c}{\partial \theta_{c6}} = {}_1^0T(\theta_{c1}) {}_2^1T(\theta_{c2}) {}_3^2T(\theta_{c3}) {}_4^3T(\theta_{c4}) {}_5^4T(\theta_{c5}) \frac{\partial {}_6^5T(\theta_{c6})}{\partial \theta_{c6}} \tag{18}$$

$$\begin{aligned} \frac{\partial {}_6^0T_c}{\partial \theta_{ck}} &= {}_1^0T(\theta_{c1}) \dots {}_{k-1}^{k-2}T\left(\theta_{c(k-1)}\right) \\ &\frac{\partial {}_k^{k-1}T(\theta_{ck})}{\partial \theta_{ck}} {}_{c(k+1)}^{k}T(\theta_{k+1}) \dots {}_6^5T(\theta_{c6}) \end{aligned} \tag{19}$$

Using equations (13), (15), (16), (17), (18), and (19), we can calculate the value of incremental joint angles $(d\theta_{c1}, d\theta_{c2}, d\theta_{c3}, d\theta_{d4}, d\theta_{c5}, d\theta_{c6})$. In order to ensure good rapidity and stability, the value of joint variables is updated according to the following principles

$$\theta'_{ck} = \begin{cases} \theta_{ck} + d\theta_{th} & \text{where} \quad d\theta_{ck} > d\theta_{th}; \\ \theta_{ck} - d\theta_{th} & \text{where} \quad d\theta_{ck} < d\theta_{th}; \\ \theta_{ck} + d\theta_{ck} & \text{else} \end{cases} \tag{20}$$

where $k = 1, 2, 3, 4, 5, 6$ and θ'_{ck} denotes the value of joint variable θ_k in the next iteration. The value of $d\theta_{th}$ is expected to match the value of η. The goal of the constant $d\theta_{th}$ is to limit the change of joint variables

$$\theta_{ck} = \theta'_{ck} \tag{21}$$

The joint variable (θ_k) is appointed a new current value (θ_{ck}) by performing equation (21), then we can perform the next round of iteration until $E \le E_0$. Flow chart of APPS for industrial robot is illustrated in Figure 3. Comparing the Figures 2 and 3, we can conclude that the mechanisms of two methods are the same.

Experiment validation

The model of industrial robot used in this article is KUKA KR 210 R2700 EXTRA as shown in Figure 4. According to the official documentation on KR 210 R2700 robot, the repeatability of KR 210 R2700 is ± 0.006 mm. However, this repeatability is really hard to achieve in actual operation. Figure 4(a) is a testbed of KR 210 R2700, and it illustrates the experiment environment of this industrial robot. Coordinate frames of KR 210 R2700 EXTRA robot are illustrated in Figure 4(b) (Table 1).

A collision-free path usually consists of a series of points in space for TCP, with the target orientation and position. The process of path tracking using APPS

Table 1. Denavit-Hartenberg parameters of KUKA KR 210 R2700 EXTRA robot.

L	θ_i	α_{i-1} (rad)	a_{i-1} (m)	d_i (m)	Range (°)
1	θ_1	π	0	−0.675	[−185, 185]
2	θ_2	$\pi/2$	0.35	0	[−95, 50]
3	θ_3	0	1.15	0	[−30, 245]
4	θ_4	$-\pi/2$	0.041	−1.2	[−350, 350]
5	θ_5	$\pi/2$	0	0	[−125, 125]
6	θ_6	$-\pi/2$	0	−0.215	[−350, 350]

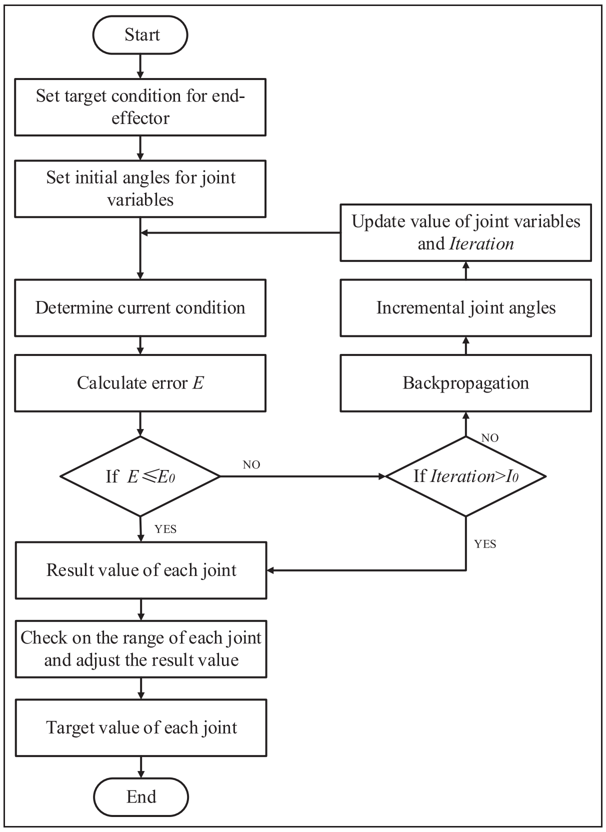
Start
Set target condition for end-effector
Set initial angles for joint variables
Update value of joint variables and Iteration
Determine current condition
Incremental joint angles
Calculate error E
Backpropagation
NO
If E≤E0
NO
If Iteration>I0
YES
YES
Result value of each joint
Check on the range of each joint and adjust the result value
Target value of each joint
End

Figure 3. Flow chart of APPS for industrial robot.

is illustrated in Figure 5, where N denotes the total number of steps. In each step, TCP of the robot manipulator is controlled to move to the next point, and the changes of joint angles are determined by APPS. A typical collision-free path for KR 210 R2700 EXTRA robot manipulator generated by high end path planner is illustrated in Figure 6,[24,25] where target point denotes target position of the TCP and initial point denotes the initial position of the TCP. In our experiment, the TCP is controlled to start from initial point, reaches all target points sequentially, and returns to initial point. The TCP is controlled to track the path in the whole experiment. This experiment can be applied to the drilling system in industrial manufacturing.

The position of TCP for each step is illustrated in Table 2, where P_I denotes the position of TCP in initial point. P_k denotes the position of TCP in target point k, where $k = 1, 2, 3, 4, 5, 6$. The X, Y, and Z denote the coordinates of TCP in Cartesian space with respect to the base frame.

In this experiment, we set target condition for the TCP by determining the elements of ${}^0_6T_d$ directly. The goal elements of ${}^0_6T_d$ for target point k are set as follows

$$ {}^0_6T_d = \begin{bmatrix} 0 & 0 & -1 & X_k \\ 0 & -1 & 0 & Y_k \\ -1 & 0 & 0 & Z_k \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{22} $$

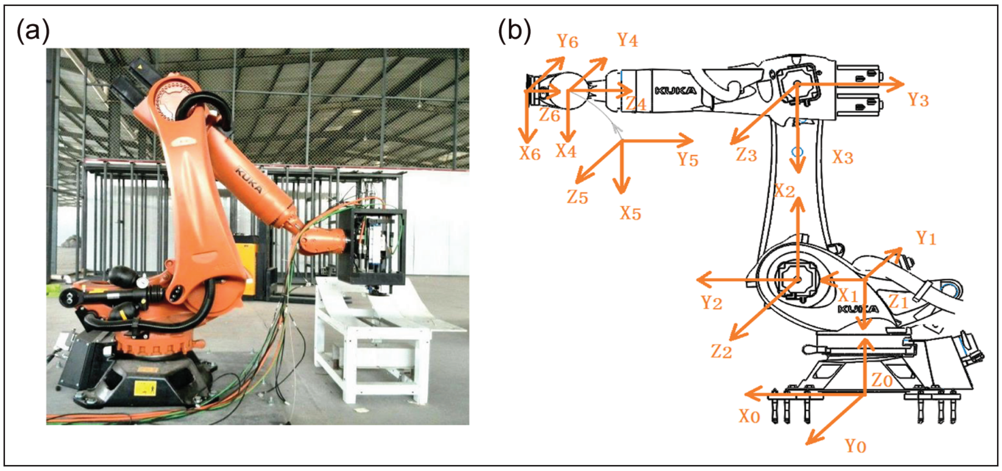

Figure 4. KUKA KR 210 R2700 EXTRA robot: (a) testbed and its experiment environment and (b) coordinate frames.

Table 2. The position of TCP for each point.

Position (m)	P_I	P_1	P_2	P_3
X_k	1.9168	1.9845	1.7850	1.6215
Y_k	0	0.6342	0.6342	0.6342
Z_k	1.5539	1.1262	1.2252	1.3242
Position (m)	P_4	P_5	P_6	P_I
X_k	1.6215	1.7850	1.9845	1.9168
Y_k	−0.6342	−0.6342	−0.6342	0
Z_k	1.3242	1.2252	1.1262	1.5539

TCP: tool center point.

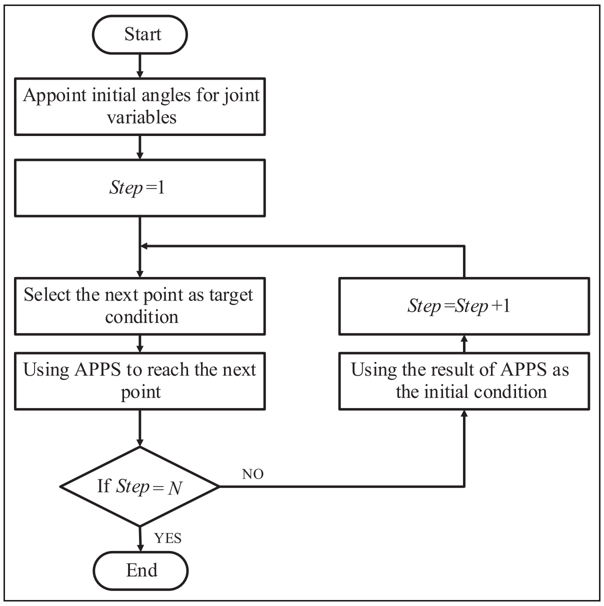
Start
Appoint initial angles for joint variables
Step=1
Select the next point as target condition
Step=Step+1
Using APPS to reach the next point
Using the result of APPS as the initial condition
If Step=N
NO
YES
End

Figure 5. Process of path tracking using APPS.

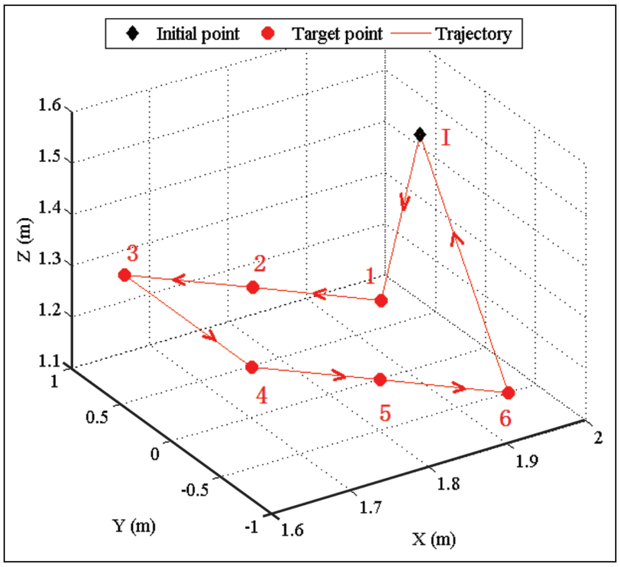

Figure 6. A typical collision-free path.

where X_k, Y_k, and Z_k are chosen from Table 2. Here, we provide experiment validation of the APPS in path tracking. The value of joint variables is illustrated in Table 3, where motion depicts the process from current condition to target condition. In Table 3, motion I-1 denotes the movement of the TCP from P_I to P_1, and motion 6-I denotes the movement of the TCP from P_6 to P_I, and motion $k-k+1$ denotes the movement of the TCP from P_k to P_{k+1}, where $k = 1, 2, 3, 4, 5$. I denotes the initial value of joint variables. R_1 represents the result of joint angles or position attained by implementing APPS.

However, the result attained by implementing APPS does not mean the result that a physical robot can actually reach. There are many issues which must be taken into consideration in actual operation of industrial robot, such as control precision for joint angles, mechanical error, and environment noise. We get the values of R_2 by rounding R_1 to two decimal places. R_2 represents the result which is actually reached by KUKA KR 210 R2700 robot. R_2/I denotes that R_2 in current motion is the I of the next motion. The motion 0 denotes the configuration of joint variables when we start the experiment, and it can be any possible configurations. Before we start the experiment, we recorded the joint angles of motion 0 ($\theta_1 = 0.01, \theta_2 = -78.33, \theta_3 = 183.51, \theta_4 = 0.16, \theta_5 = -44.87, \theta_6 = -179.93$). In addition, the other parameters are selected as $\eta = 0.01$, $d\theta_{th} = \pi/180$, $E_0 = 10^{-8}$, and $I_0 = 1000$. The experiment result shows that this set of η, $d\theta_{th}$, E_0, and I_0 have a good performance.

The processes of motion I-1, 3-4, and 6-I are presented in Supplementary Materials. Additionally, the motion introduced here does not represent the movement of physical robot, and they denote the virtual process in solving IK problem. The actual movement of industrial robot should be planned by high end path planner.[24,25]

The position of TCP with respect to the base frame is illustrated in Table 4. T denotes the target value according to Table 2. We introduce mean absolute

Table 3. Configuration and results of APPS for joint variables.

Motion	Motion 0	Motion 1-1		Motion 1-2		Motion 2-3	
J (°)	I	R_1	R_2/I	R_1	R_2/I	R_1	R_2/I
θ_1	0.01	−20.0995	−20.10	−21.9957	−22.00	−24.2704	−24.27
θ_2	−78.33	−66.5326	−66.53	−75.7315	−75.73	−85.0300	−85.03
θ_3	183.51	184.7585	184.76	191.8096	191.81	197.4984	197.50
θ_4	0.16	142.2740	142.27	137.4221	137.42	130.2858	130.29
θ_5	−44.87	34.1622	34.16	33.6108	33.61	32.6040	32.60
θ_6	−179.93	−147.3760	−147.38	−142.576	−142.58	−135.1768	−135.18

Motion	Motion 3-4		Motion 4-5		Motion 5-6		Motion 6-1	
J (°)	R_1	R_2/I	R_1	R_2/I	R_1	R_2/I	R_1	R_2/I
θ_1	24.2704	24.27	21.9967	22.00	20.0955	20.10	0.0122	0.01
θ_2	−85.0300	−85.03	−75.7318	−75.73	−66.5326	−66.53	−78.3288	−78.33
θ_3	197.4985	197.50	191.8080	191.81	184.7585	184.76	183.5099	183.51
θ_4	229.7142	229.71	222.5843	222.58	217.7259	217.73	0.1615	0.16
θ_5	32.6040	32.60	33.6099	33.61	34.1623	34.16	−44.8711	−44.87
θ_6	−224.8230	−224.82	−217.4305	−217.43	−212.6243	−212.62	−179.9318	−179.93

APPS: autonomous path planning solution.

Table 4. Position of TCP in each motion.

	Position (m)	Motion 1-1	Motion 1-2	Motion 2-3	Motion 3-4	Motion 4-5	Motion 5-6	Motion 6-1
T	X	1.9485	1.7850	1.6215	1.6215	1.7850	1.9485	1.9168
	Y	0.6342	0.6342	0.6342	−0.6342	−0.6342	−0.6342	0
	Z	1.1262	1.2252	1.3242	1.3242	1.2252	1.1262	1.5539
R_1	X	1.94850	1.78500	1.62150	1.62150	1.78500	1.94850	1.91680
	Y	0.63421	0.63419	0.63419	−0.63419	−0.63421	−0.63421	0
	Z	1.12621	1.22518	1.32418	1.32418	1.22522	1.12621	1.55390
R_2	X	1.94846	1.78496	1.62150	1.62150	1.78496	1.94846	1.91678
	Y	0.63437	0.63432	0.63419	−0.63419	−0.63432	−0.63437	0.00009
	Z	1.12609	1.22513	1.32414	1.32414	1.22513	1.12609	1.55393

TCP: tool center point.

Table 5. MAE of position for TCP.

MAE (mm)	Motion 1-1	Motion 1-2	Motion 2-3	Motion 3-4	Motion 4-5	Motion 5-6	Motion 6-1
MAE_1	0.01	0.01	0.01	0.01	0.01	0.01	0
MAE_2	0.11	0.07	0.01	0.01	0.08	0.11	0.05
MAE_S	0.12	0.08	0.02	0.02	0.09	0.12	0.05

MAE: mean absolute error; TCP: tool center point.

error (MAE) of position as evaluation index for the performance of APPS (Table 5). MAE_1 denotes the MAE between T and R_1 in a motion, and it means the error introduced by APPS. MAE_2 denotes the MAE between R_1 and R_2 in a motion, and it means the error introduced by physical industrial robot. MAE_S denotes the maximum possible MAE between T and R_2 in a motion, and it means the error in the whole system. MAE_S is calculated as follows

$$MAE_S = MAE_1 + MAE_2 \qquad (23)$$

Thus, we can conclude that the APPS converges to target solution very well, with a very tiny TCP position MAE of 0.12, 0.08, 0.02, 0.02, 0.09, 0.12, and 0.05 mm in each step (Table 4). The MAE in this system is illustrated in Figure 7.

In this experiment validation, we controlled the KUKA KR robot to track a collision-free path using APPS algorithm. The MAE is used as evaluation index

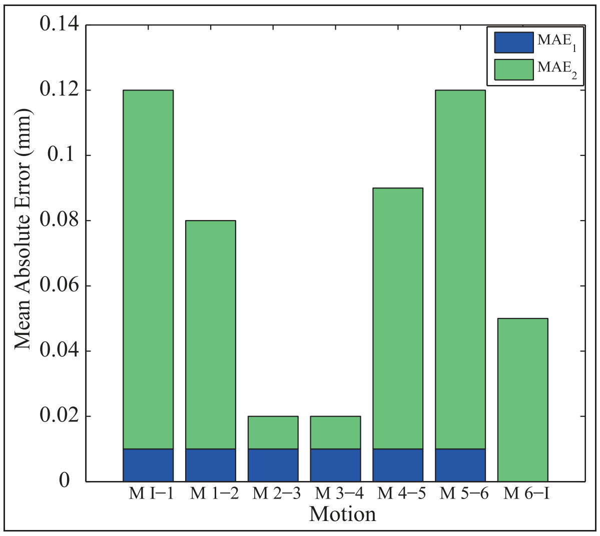

Figure 7. Mean absolute error of position for TCP.

Table 6. Comparison between APPS and NR.

Algorithm			
Motion	MAE (mm)	Time (s)	Iteration
APPS			
Motion I-1	0.01	16.36182	438
Motion 1-2	0.01	10.00347	256
Motion 2-3	0.01	12.62770	316
Motion 3-4	0.01	22.45021	618
Motion 4-5	0.01	10.69781	278
Motion 5-6	0.01	9.10838	233
Motion 6-I	0	11.31745	281
NR			
Motion I-1	FALSE		
Motion 1-2	2.00	24.78871	4000
Motion 2-3	1.40	24.95875	4000
Motion 3-4	59.53	25.24718	4000
Motion 4-5	0.83	19.11921	4000
Motion 5-6	FALSE		
Motion 6-I	204.5	19.42067	4000

MAE: mean absolute error; APPS: autonomous path planning solution; NR: Newton–Raphson.

for each motion. This experiment validation proved that the MAE introduced by APPS is very tiny. The learning rate is a main influence factor to the performance of the APPS. Learning rate has a strong correlation to both computation time and convergence of APPS algorithm. We have performed sufficient experiments to determine the best value for the learning rate.

Comparison between APPS and NR

In order to show the accuracy and efficiency of APPS, comparison between APPS and the numerical methods based on NR method in path tracking is provided. As it is known to all, the numerical methods based on NR method requires six initial guess joint angles, and it would converges to the target value faster if the initial guess angles are accurate. Both APPS and NR are performed on an Intel i5 2.67 GHz with 4 GB RAM using MATLAB software program. The NR method is achieved using the robotics toolbox, which is a MATLAB-based toolbox. The toolbox provides many functions that are useful to the study of manipulators such as kinematics, dynamics, and path planning.[26] It is important to note that we use the modified DH parameter when constructing the calculation model of manipulator.

The computation time and iteration of motion given in Table 4 are illustrated in Table 6. The mean MAE of APPS is smaller (0.01 mm) in comparison to NR method (53.65 mm). The mean calculation time of APPS is also smaller (13.2238 s) in comparison to NR method (22.7067 s). The APPS method has many other advantages in comparison to NR method. First, we can conclude that the APPS performs well when singularity exists, such as in the process of motions I-1 and 5-6. The NR method cannot solve the IK problem in the process of motions 3-4 and 6-I, where the FALSE presents that NR algorithm fails to converge because of singularity. Second, the APPS does not need any accurate initial guess angles, while NR cannot converge to target values with inaccurate initial guess angles, such as in the process of motions 3-4 and 6-I. In the process of motion 3-4, the changes of each joint angle are 48.5404, 0, −0.0015, 99.4242, 0.0040, and −86.6430. The MAE of APPS is 0.01 mm in motion 3-4, while the MAE of NR is 59.53 mm. In the process of motion 6-I, the changes of each joint angle are −20.0878, −11.7988, −1.2501, 142.4315, −79.0311, and 32.6882. The MAE of APPS is 0 mm in motion 6-I, while the MAE of NR is 204.50 mm. In addition, the motions of I-1 and 6-I are two complex motions for the manipulator because the position and orientation of TCP are both changed. The MAE of APPS in motions I-1 and 6-I are 0.01 and 0, respectively. However, the NR method failed to converge in motion I-1, and the MAE of NR is 204.50 mm in motion 6-I.

Conclusion

This article presented an autonomous inverse kinematic solution APPS to solve IK problem. The APPS is based on BP algorithm and it is an iterative solution. The experiment results prove that the APPS has a good performance compared with NR solution. As a numerical method, APPS has the following features: (1) it does not need an accurate initial angle guess, (2) it has low computational complexity and less number of iterations, and (3) it has high accuracy in path tracking. This study focused on determining the changes of joint angles in solving IK problem, and the dynamics problem and real-time control would be considered in the further research.

Acknowledgements

The participants in this research are all thankworthy for their contribution to the research.

Declaration of conflicting interests

The author(s) declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

Funding

The author(s) disclosed receipt of the following financial support for the research, authorship, and/or publication of this article: This work was supported by the National Natural Science Foundation of China (No. 61375085).

References

1. Fuentes CT and Bastian AJ. Where is your arm? Variations in proprioception across space and tasks. *J Neurophysiol* 2010; 103: 164–171.
2. Adamovich SV, Berkinblit MB, Fookson O, et al. Pointing in 3D space to remembered targets. I. Kinesthetic versus visual target presentation. *J Neurophysiol* 1998; 79: 2833–2846.
3. Kucuk S and Bingul Z. Inverse kinematics solutions for industrial robot manipulators with offset wrists. *Appl Math Model* 2014; 38: 1983–1999.
4. Manocha D and Canny J. Efficient inverse kinematics for general 6R manipulators. *IEEE Trans Robot Automat* 1994; 10: 648–657.
5. Raghavan M and Roth B. Inverse kinematics of the general 6R manipulator and the related linkages. *Trans ASME: J Mech Des* 1993; 115: 502–508.
6. Featherstone R. Position and velocity transformations between robot end-effector coordinates and joint angles. *Int J Robot Res* 1983; 2: 135–145.
7. Uicker JJ, Denavit J and Hartenberg RS. An iterative method for the displacement analysis of spatial mechanisms. *ASME J: Appl Mech* 1964; 21: 309–314.
8. Goldenberg AA, Apkarian JA and Smith HW. A new approach to kinematic control of robot manipulators. *ASME J: Dyn Syst Meas Control* 1987; 109: 97–103.
9. Szkodny T. Avoiding of the singularities of contemporary industrial robots. *Lect Note Comput Sci* 2014; 8918: 183–194.
10. Wang L-CT and Chen CC. A combined optimization method for solving the inverse kinematics problem of mechanical manipulators. *IEEE Trans Robot Automat* 1991; 7: 489–499.
11. Hasan AT, Ismail N, Hamouda AMS, et al. Artificial neural network-based kinematics Jacobian solution for serial manipulator passing through singular configurations. *Adv Eng Softw* 2010; 41: 359–367.
12. Hasan AT, Hamouda AMS, Ismail N, et al. An adaptive-learning algorithm to solve the inverse kinematics problem of a 6 D.O.F serial robot manipulator. *Adv Eng Softw* 2006; 37: 432–438.
13. Dong Y, He W and Sun C. Adaptive neural impedance control of a robotic manipulator with input saturation. *IEEE Trans Syst Man Cybernet* 2015; 99: 1–11.
14. Li Z, Yang C, Su C-Y, et al. Decentralized fuzzy control of multiple cooperating robotic manipulators with impedance interaction. *IEEE Trans Fuzzy Syst* 2015; 23: 1044–1056.
15. Rasit K. A genetic algorithm approach to a neural-network-based inverse kinematics solution of robotic manipulators based on error minimization. *Inform Sci* 2013; 222: 528–543.
16. Wei Y, Jian S, He S, et al. General approach for inverse kinematics of nR robots. *Mech Mach Theory* 2014; 75: 97–106.
17. Qiao S, Liao Q, Wei S, et al. Inverse kinematic analysis of the general 6R serial manipulators based on double quaternions. *Mech Mach Theory* 2010; 45: 193–199.
18. Rudny T. Solving inverse kinematics by fully automated planar curves intersecting. *Mech Mach Theory* 2014; 74: 310–318.
19. Su C-Y, Li Z and Li G. Fuzzy approximation-based adaptive backstepping control of an exoskeleton for human upper limbs. *IEEE Trans Fuzzy Syst* 2015; 23: 555–566.
20. Jiang D, Zhao Z, Xu Z, et al. How to reconstruct end-to-end traffic based on time-frequency analysis and artificial neural network. *AEU: Int J Electr Commun* 2014; 68: 915–925.
21. Nedic V, Despotovic D, Cvetanovic S, et al. Comparison of classical statistical methods and artificial neural network in traffic noise prediction. *Environ Impact Assess Rev* 2014; 49: 24–30.
22. Arnone E, Francipane A, Noto LV, et al. Strategies investigation in using artificial neural network for landslide susceptibility mapping: application to a Sicilian catchment. *Journal of Hydroinform* 2014; 16: 502–515.
23. Rodrigues PL, Rodrigues NF, Pinho AC, et al. Automatic modeling of pectus excavatum corrective prosthesis using artificial neural networks. *Med Eng Phys* 2014; 36: 1338–1345.
24. Gasparetto A and Zanotto V. Optimal trajectory planning for industrial robots. *Adv Eng Softw* 2010; 41: 548–556.
25. Rubio FJ, Valero FJ, Suner JL, et al. Simultaneous algorithm to solve the trajectory planning problem. *Mech Mach Theory* 2009; 44: 1910–1922.
26. Corke PI. *Robotics, vision & control: fundamental algorithms in MATLAB*. Berlin: Springer, 2011.