Proceedings of the 2009 IEEE International Conference on Systems, Man, and Cybernetics
San Antonio, TX, USA - October 2009

Motion Planning with Obstacle Avoidance for Kinematically Redundant Manipulators Based on Two Recurrent Neural Networks

Xiaolin Hu, Jun Wang, and Bo Zhang

Abstract— **Inverse kinematic motion planning of redundant manipulators by using recurrent neural networks in the presence of obstacles and uncertainties is a real-time nonlinear optimization problem. To tackle this problem, two subproblems should be resolved in real time. One is the determination of critical points on a given manipulator closest to obstacles, and the other is the computation of joint velocities of the manipulator which can direct the manipulator following a desired trajectory and away from obstacles if it is getting close to them. Different from our previous approaches where the critical points on the manipulator were assumed to be known, these points are to be computed by using a recurrent neural network in the paper. A time-varying quadratic programming problem is formulated for avoiding polyhedral obstacles. In view that the problem is not strictly convex, an existing recurrent neural network, general projection neural network, is applied for solving it. By introducing a velocity smoothing technique into our previous quadratic programming formulation of the joint velocity assignment problem, a recently developed recurrent neural network, improved dual neural network, is proposed to solve it, which features lower structural complexity compared with existing neural networks. Moreover, The effectiveness of the proposed neural networks is demonstrated by simulations on the Mitsubishi PA10-7C manipulator.**

I. Introduction

Kinematically redundant robot manipulators are those having more degrees of freedom than required to perform given end-effector moving tasks. Many studies have been reported about using kinematically redundant manipulators for applications [1]. For obstacle avoidance, it is often demanded to determine the critical point on the manipulator that has the minimum (or near minimum) distance to obstacles. However, in some recent studies [2], [3] the critical points were assumed to be known *a priori*. This assumption oversimplified real situations. Indeed, even if the obstacles are static in the workspace, the nearest points may change positions on the surfaces when the manipulator is moving. In sensor-based motion planning, the point can be determined by synthetic information of sensor fusion technology; e.g., utilizing vision, ultrasonic and infra-red sensors [4], [5]; while in model-based motion planning, it can be computed via online distance minimization between the manipulator and the obstacles. In this paper, the obstacles are assumed to be 3D convex polyhedron and a time-varying convex quadratic programming (QP) problem is formulated to determine the critical point.

Obtaining the critical points is just a prerequisite. The next step is to direct the manipulator away from the obstacles. Since serial-link manipulators are considered throughout the paper, every link should be assigned a velocity under geometric configuration constraints. Many strategies about the velocity control have been reported. A popular approach is to solve a formulated optimization problem [2], [6], [7]. This approach has several advantages including practical considerations such as joint limits, joint velocity limits and obstacle avoidance into the optimization problem as constraints. In particular, a QP formulation was proposed for obstacle avoidance in [2], and an improved formulation was presented in [3]. However, since a fixed parameter was used in a critical inequality constraint in [3], an abrupt behavior of the manipulator velocity was observed, which introduced risk of derailing the operation. In the paper, this hard constraint is replaced by a soft constraint using a smoothing technique. The troublesome phenomenon is then eliminated.

It is seen that the entire control scheme proposed in the paper comprises solving two time-varying QP problems, which are called *critical point problem* and *joint velocity problem*.

The bottleneck of any kinematic control algorithm is intensive computation of manipulator motion in real-time. To deal with this difficulty, recurrent neural networks were developed for dynamic optimization and appeared to be a promising approach. In the last decades, many recurrent neural networks have been developed and applied focusing on achieving pseudoinverse solutions [8], minimum torque [9], [10], minimum kinetic energy [11], optimal grasping force [12], and so on. In [2] and [3], a recurrent neural network called *simplified dual neural network (SDNN)* reported in [13] was applied for solving the joint velocity problem. In this paper, an even simpler neural network, the *improved dual neural network (IDNN)* reported in [14] is applied for solving the problem. Since neither SDNN nor IDNN can solve the critical point problem which is not strictly convex, a *general projection neural network (GPNN)* [15] is applied.

The work described in the paper was supported by the Research Grants Council of the Hong Kong Special Administrative Region, China, under Grants G_HK010/06 and CUHK417608E, by the Science and Technology Commission of Shanghai Municipality under Grant 08JC1412100, by the National Natural Science Foundation of China under Grants 60805023, 60621062 and 60605003, by the National Key Foundation R&D Project under Grants 2003CB317007, 2004CB318108 and 2007CB311003, by the China Postdoctoral Science Foundation under Grants 20080430032 and 200801072, and by the Basic Research Foundation of Tsinghua National Laboratory for Information Science and Technology (TNList).

Xiaolin Hu and Bo Zhang are with State Key Laboratory of Intelligent Technology and Systems, Tsinghua National Laboratory for Information Science and Technology (TNList), and Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China xiaolin.hu@gmail.com; dcszb@tsinghua.edu.cn

Jun Wang is with Department of Mechanical and Automation Engineering, The Chinese University of Hong Kong, Shatin, N.T., Hong Kong jwang@mae.cuhk.edu.hk

978-1-4244-2794-9/09/$25.00 ©2009 IEEE

Then, the inverse kinematic motion planning problem can be solved by coupling two recurrent neural networks, IDNN and GPNN. Simulation results will be presented to validate this approach.

II. Obstacle avoidance scheme

A. Critical Point

An obstacle avoidance task is to identify the critical point on the manipulator, from time to time, that has the minimum distance to obstacles, then assign desired joint velocities to the manipulator and direct it away from the obstacles. At any time, there is a point O on every obstacle, defined as *obstacle point*, and a point C on a link, defined as *critical point*, so that $|OC|$ is the minimum distance between the particular obstacle-link pair. If the obstacle point O is known at anytime, it is easy to locate the critical point C on the link This is the case considered in [2] and [3]. Actually, in those two studies, an obstacle was represented by a point without geometric size. As discussed in introduction, this assumption oversimplified real situations.

We now discuss how to obtain the critical point C on manipulator and the corresponding point O on these objects in a model-based environment where obstacles are 3D bodies. The problem is formulated as a quadratic optimization problem:

$$\begin{array}{ll}\text{minimize} & \frac{1}{2}\|(x,y,z)^T-(\tilde{x},\tilde{y},\tilde{z})^T\|_2^2\\ \text{subject to} & (x,y,z)^T\in\mathcal{U},(\tilde{x},\tilde{y},\tilde{z})^T\in\mathcal{V}\end{array}$$

where $\mathcal{U}$ and $\mathcal{V}$ stand for the subsets of $\Re^3$ occupied by the obstacle and the manipulator, respectively. For every link and every obstacle we should seek the optimum $(x^*,y^*,z^*)^T$ and $(\tilde{x}^*,\tilde{y}^*,\tilde{z}^*)^T$ to the above problem, which represent the position of O on the obstacle and the critical point of C on the link, respectively. For simplicity, throughout the paper, the links are approximated by line segments and the detailed shape information (such as radius, thickness) is ignored. But in principle, more realistic configurations of links can be considered. Then for every link we have

$$\mathcal{V}=\left\{(\tilde{x},\tilde{y},\tilde{z})^T\in\Re^3\;\middle|\;\begin{array}{l}G\cdot(\tilde{x},\tilde{y},\tilde{z})^T=d,\\ \underline{\eta}\le(\tilde{x},\tilde{y},\tilde{z})^T\le\overline{\eta}\end{array}\right\},$$

where

$$G=\begin{pmatrix}y_2-y_1 & x_1-x_2 & 0\\ z_2-z_1 & 0 & x_1-x_2\end{pmatrix},$$
$$d=\begin{pmatrix}(y_2-y_1)x_1-(x_2-x_1)y_1\\ (z_2-z_1)x_1-(x_2-x_1)z_1\end{pmatrix}.$$

In above description, $(x_1,y_1,z_1)^T$ and $(x_2,y_2,z_2)^T$ stand for the position vectors of the two ends of the link, and $\eta,\overline{\eta}\in\Re^3$ stand for the lower and upper bounds of $(\tilde{x},\tilde{y},\tilde{z})^T$. If $\mathcal{U}$ is a convex polyhedron, without loss of generality, it can be written as

$$\mathcal{U}=\{(x,y,z)^T\in\Re^3|A\cdot(x,y,z)^T\le e\}$$

where $A\in\Re^{q\times3}$ and $e\in\Re^q$ are constants. In this case, the optimization problem becomes a (convex) QP problem. By introducing a new variable $w=(x^T,y^T,z^T,\tilde{x}^T,\tilde{y}^T,\tilde{z}^T)^T$, the problem can be put into the following compact form,

$$\begin{array}{ll}\text{minimize} & \frac{1}{2}w^TQw\\ \text{subject to} & Kw\in\Pi,w\in\mathcal{W}\end{array}\qquad(1)$$

where

$$Q=\begin{pmatrix}I & -I\\ -I & I\end{pmatrix},K=\begin{pmatrix}A & 0\\ 0 & G\end{pmatrix},$$
$$\Pi=\left\{\nu\in\Re^{q+2}\;\middle|\;\begin{pmatrix}-\infty\\ d\end{pmatrix}\le\nu\le\begin{pmatrix}e\\ d\end{pmatrix}\right\},$$
$$\mathcal{W}=\left\{w\in\Re^6\;\middle|\;\begin{pmatrix}-\infty\\ \underline{\eta}\end{pmatrix}\le w\le\begin{pmatrix}\infty\\ \overline{\eta}\end{pmatrix}\right\}.$$

By solving (1), the obstacle point O, critical point C, together with the distance between them can be obtained simultaneously. Note that (1) is a time-varying optimization problem as the the parameters $K,\Pi,\mathcal{W}$ change with time while the manipulator or the obstacle itself is moving.

Remark 1: The convex optimization problem (1) is established for every link of the manipulator versus every obstacle. For the case of multiple links or obstacles, multiple optimization problems in the form of (1) should be present. In this sense, it is a local strategy, not a global one.

B. Joint Velocity

Taken obstacle avoidance into consideration, the forward kinematics of a serial-link manipulator can be described by the following augmented forward kinematics equation:

$$r_e(t)=f_e(\theta(t)),\quad r_c(t)=f_c(\theta(t),t)\qquad(2)$$

where $r_e\in\Re^m$ is m-dimensional position and/or orientation vector of the end-effector, $r_c\in\Re^3$ is the position vector of a critical point on every link, $\theta(t)$ is the joint vector of the manipulator, and f_e and f_c are nonlinear functions of the manipulator with respect to the end-effector and the critical point respectively. Note that the critical point can vary its position on a link and that is why $r_c(t)$ depends also on t. There always exist more than one critical point in the system no matter there are multiple obstacles or not because a manipulator always has multiple links. Therefore multiple equations similar to the second one should be present.

The manipulator path planning problem (also called inverse kinematics problem or kinematic control problem) is to find the joint variable $\theta(t)$ for any given $r_e(t)$ and $r_c(t)$ through the inverse mapping of (2). Unfortunately, it is usually impossible to find an analytic solution due to the nonlinearity of $f_e(\cdot)$ and $f_c(\cdot)$. The inverse kinematics problem is thus usually solved at the velocity level with the relation

$$J_e(\theta)\dot{\theta}=\dot{r}_e,\quad J_c(\theta)\dot{\theta}\approx\dot{r}_c,\qquad(3)$$

where $J_e(\theta)=\partial f_e(\theta)/\partial\theta$, $J_c(\theta)=\partial f_c(\theta)/\partial\theta$ are Jacobian matrices, $\dot{r}_e$ is the desired velocity of the end-effector, and $\dot{r}_c$ is the desired velocity of the critical point which should be properly selected to effectively direct the link away from the obstacle. Here, the dependence of r_c on time is ignored for simplifying the problem. In other words, it is assumed that the sliding velocity of the critical point on a link is negligible

compared with the joint velocities. Simulation studies (see Section IV) have shown that this simplification is acceptable in terms of the task competence and tracking error. However, in practice it is often difficult to determine the suitable magnitude of escape velocity $\dot{r}_c$. Even worse, if there are p critical points, we should have $3p$ equalities of $J_c(\theta)\dot{\theta} = \dot{r}_c$; then, if $3p > n$, these equalities will be overdetermined. Because of these considerations, $J_c(\theta)\dot{\theta} = \dot{r}_c$ in (3) was replaced by inequality constraints [2], [3]. the formulated inequality in [3] is

$$\overrightarrow{OC}^T J_c(\theta)\dot{\theta} \geqslant c, \tag{4}$$

where $\overrightarrow{OC}$ represent the position vector pointing from O to C (see Fig. **??**) and c is a positive constant. It was shown that this formulation enlarges the feasible solution space of the QP problem in [2].

In order not to introduce too much restriction on the feasible space and consequently obtain a better *optimum*, the inequality constraint (4) should be applied only when $|\overrightarrow{OC}|$ is less than some threshold. However, such an idea suffers from discontinuity in joint velocities; see the simulation example in [3] and note an abrupt change in curves in Fig. 7. To overcome this difficulty, here a time-varying parameter is introduced

$$\hat{b}(t) = s(|\overrightarrow{OC}|)\max(0, -\overrightarrow{OC}^T J_c(\theta)\dot{\theta})$$

where $s(d)$ is a distance-based smoothing function defined as

$$s(d) = \begin{cases} 1, & \text{if } d \geqslant d_2 \\ \sin^2\left(\frac{\pi}{2}\frac{d-d_1}{d_2-d_1}\right), & \text{if } d_1 < d < d_2 \\ 0, & \text{if } d \leqslant d_1 \end{cases} \tag{5}$$

with d_1, d_2 standing for the inner safety threshold and outer safety threshold, respectively ($d_2 > d_1$), provided by users. Then (4) is replaced with

$$-\overrightarrow{OC}^T J_c(\theta)\dot{\theta} \leqslant \hat{b}(t). \tag{6}$$

When the distance between the link and obstacle reaches d_2, the system begins to repel the link away in a more and more stringent manner. When the distance reaches d_1, the system prevents the link getting closer to the obstacle [2].

In view that $\overrightarrow{OC} = (\ x_c - x_o, \quad y_c - y_o, \quad z_c - z_o\ )^T$, (6) can be rewritten as

$$-(\ x_c - x_o, \quad y_c - y_o, \quad z_c - z_o\ )J_c(\theta)\dot{\theta} \leqslant \hat{b}(t).$$

Suppose there are p critical points. Let $b = (\hat{b}, \cdots, \hat{b})^T \in \Re^p$ and

$$L(\theta) = \begin{pmatrix} (\ x_{o_1} - x_{c_1}, \quad y_{o_1} - y_{c_1}, \quad z_{o_1} - z_{c_1}\ )J_{c_1}(\theta) \\ \vdots \\ (\ x_{o_p} - x_{c_p}, \quad y_{o_p} - y_{c_p}, \quad z_{o_p} - z_{c_p}\ )J_{c_p}(\theta) \end{pmatrix},$$

where the subscripts o_i and c_i constitute an obstacle-point-critical-point pair. For example, if there are two obstacles and three serial links, then $p = 6$ because each obstacle corresponds to a critical point on each link. By these definitions, the above inequality becomes

$$L(\theta(t))\dot{\theta}(t) \leqslant b(t). \tag{7}$$

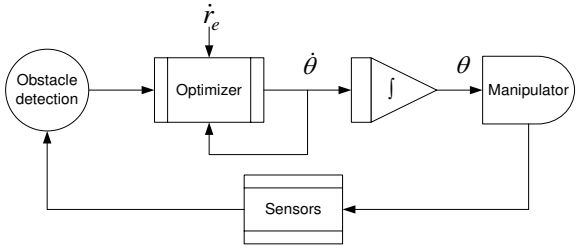


Fig. 1. Kinematic control process.

Similar to [3] the following optimization problem is formulated:

$$\begin{aligned} \text{minimize} \quad & \frac{1}{2}\|\dot{\theta}(t)\|_2^2 \\ \text{subject to} \quad & J_e(\theta(t))\dot{\theta}(t) = \dot{r}_e(t), \\ & L(\theta(t))\dot{\theta}(t) \leqslant b(t), \\ & l \leqslant \dot{\theta}(t) \leqslant h, \end{aligned} \tag{8}$$

where l and h are respectively lower and upper bounds of the joint velocity.

Like problem (1), problem (8) is also a time-varying QP problem. The entire kinematic control process is shown in Fig. 1, similar to most optimization-based kinematic control approaches [2], [6], [7], [9], [16], [17]. The desired end-effector velocity $\dot{r}_e$ and the obstacle information are fed into the optimizer as inputs. In sensor-based control, the critical points and obstacle points are detected with respect to the current pose of manipulator; and in model-based control, they are determined by solving problem (1). The time-varying parameters of problems (1) and (8) are determined by the pose of the manipulator and the position of the obstacle. The optimal joint rate $\dot{\theta}$ that could make the manipulator avoid obstacles is generated as the output of the "Optimizer" block. By taking integration of the joint velocities with the known initial values, the pose of the manipulator is determined.

III. Neural network models

The bottleneck of the proposed method in Fig. 1 is the intensive computation for solving the QP problems (1) and (8), i.e., the "Obstacle detection" block and "Optimizer" block. If conventional numerical algorithms are used, two processing units with very high performance are required. We here suggest two neural networks for solving (1) and (8), which can be realized by hardware systems connected by dense interconnections of simple analog computational elements. Because of the analog nature of circuit systems, neural networks have great potential in handling computationally expensive problems on-line [8]–[12].

Before the presentation of the neural networks, an activation function is introduced first. Suppose $\mathcal{X}$ is a box set, i.e., $\mathcal{X} = \{x \in \Re^n | \underline{x}_i \leq x_i \leq \overline{x}_i, \forall i = 1, ..., n\}$, then $P_\Omega(x) = (P_\Omega(x_1), \cdots, P_\Omega(x_n))^T$ with

$$P_\Omega(x_i) = \begin{cases} \underline{x}_i, & x_i < \underline{x}_i, \\ x_i, & \underline{x}_i \leq x_i \leq \overline{x}_i, \\ \overline{x}_i, & x_i > \overline{x}_i. \end{cases} \tag{9}$$

Note $\overline{x}_i$ can be $+\infty$ and $\underline{x}_i$ can be $-\infty$. In particular, if $\underline{x} = 0$ and $\overline{x} = \infty$, the operator becomes $P_{\Re^n_+}(x)$, where $\Re^n_+$ denotes the nonnegative quadrant of $\Re^n$. To simplify the notation, in this case it is written as x^+. And the definition can be simplified as $x^+ = (x_1^+, \cdots, x_n^+)^T$ with $x_i^+ = \max(x_i, 0)$.

Consider solving problem (1). Because Q is positive semidefinite only and many neural networks including the SDNN used in [3] can not be applied. But according to [18], this problem can be formulated into an equivalent generalized linear variational inequality, and as a consequence, it can be solved by the general projection neural network (GPNN) studied extensively in [14], [15], [19]. When specialized to solve (1) the GPNN's dynamic equation is as follows (cf. eqn. (17) in [18]):

$$\epsilon \frac{du}{dt} = (M+N)^T\{-Nu + P_{\mathcal{W}\times\Pi}((N-M)u)\}, \quad (10)$$

where $\epsilon > 0$ and $u = (w^T, \nu^T)^T$ is the state variable and

$$M = \begin{pmatrix} Q & -K^T \\ 0 & I \end{pmatrix}, N = \begin{pmatrix} I & 0 \\ K & 0 \end{pmatrix}.$$

The output of the neural network is $w(t)$, which is part of the state vector $u(t)$. The architecture of the neural network can be drawn similarly as in [19] or [15], which is omitted here for brevity. Since Q is positive semidefinite, according to [18], the neural network is globally asymptotically convergent to a solution of (1).

For solving (8) there exists many recurrent neural networks, among which the SDNN devised in [13] was the simplest, as discussed in [3]. By observing the special form of the problem, in the paper we propose to use the improved dual neural network (IDNN) which was recently proposed in [20]:

- State equation

$$\epsilon \frac{d}{dt}\begin{pmatrix} \varrho \\ \varsigma \end{pmatrix} = -\begin{pmatrix} \varrho - (\varrho + L(\theta)v - b)^+ \\ Je(\theta)v - \dot{r}_e \end{pmatrix}, \quad (11a)$$

- Output equation

$$v = P_\Omega(-L(\theta)^T\varrho + Je(\theta)^T\varsigma), \quad (11b)$$

where $\epsilon > 0$ is a constant scaling factor, $\Omega = [l, h]$ is a box set, $(\varrho^T, \varsigma^T)^T$ is the state vector and v is the output vector $\dot{\theta}$. To realize this network by analogy circuits, $n+p$ amplifiers, $m+p$ integrators and $O[n(m+p)]$ connections are needed. According to [20], the neural network is globally asymptotically convergent to a solution of (8).

Note that the above network is even simpler than the SDNN adopted in [3]. Actually, if that model is used to solve (8), the number of amplifiers and the number of integrators will be both $p+n$, while the number of connections will be in the order of $O[(p+n)^2 + n(p+n)]$.

IV. Simulation results

In this section, the validity of the proposed obstacle avoidance scheme and the real-time solution capabilities of the proposed neural networks GPNN and IDNN are shown through simulations with the Mitsubishi 7-DOF PA10-7C manipulator. The coordinates setup, structure parameters, joint limits, and joint velocity limits of this manipulator can be found in [9]. In this study, only the position of the end-point is concerned, then $m = 3$ and $n = 7$. The scaling factors ϵ that control the convergence rates of the neural networks (10) and (11) are both set to 10^{-4}. The inner and outer safety threshold d_1 and d_2 in (5) are set to 0.05m and 0.1m, respectively.

Let the end-effector follow a circular trajectory described by

$$\begin{cases} x_e(t) = x_0 - 0.2 + 0.2\cos(2\pi \sin^2 \frac{\pi t}{20}) \\ y_e(t) = y_0 + 0.2\sin(2\pi \sin^2 \frac{\pi t}{20})\cos\frac{\pi}{6} \\ z_e(t) = z_0 + 0.2\sin(2\pi \sin^2 \frac{\pi t}{20})\sin\frac{\pi}{6}. \end{cases}$$

The radius is 20cm. The task time of the motion is 10s and the initial joint angles are setup as $\theta(0)$=(0, $-\pi/4$, 0, $\pi/2$, 0, $-\pi/4$, 0). Then the initial position of the end-effector (x_0, y_0, z_0) can be calculated according to the configuration of the manipulator. The desired velocity $\dot{r}_e(t)$ is determined by taking the derivatives of $x_e(t), y_e(t), z_e(t)$ with respect to t. In the workspace, suppose that there is a moving tetrahedral obstacle when the manipulator is accomplishing the motion task. Let the obstacle move also along a circular path, which is described by

$$\begin{cases} x_o(t) = x_0 - 0.3\sin(\frac{\pi t}{10}) \\ y_o(t) = y_0 - 0.3 + 0.3\cos(\frac{\pi t}{10}) \\ z_o(t) = z_0, \end{cases}$$

where $(x_o, y_o, z_o)^T$ is any point in the obstacle and $(x_0, y_0, z_0)^T$ is the initial position of that point. Clearly, the path is a half circle with radius 30cm parallel to the x-y plane. The initial positions of the tetrahedron vertices are respectively (-0.4m, 0,0.2m), (-0.3m,0,0.2m), (-0.35m,-0.1m,0.2m), (-0.35m, -0.05m, 0.4m). By some basic calculations, the parameters in (1) can be determined easily. Fig. 2 illustrates the moving trajectory of the obstacle as well as the desired path of the end-effector.

We simultaneously solve problems (1) and (8) by coupling the GPNN (10) and IDNN (11). Fig. 3 shows the continuous solution $\dot{\theta}(t)$ to problem (8). By virtue of the solution of problem (1) obtained by the GPNN (10) it is easily to calculate the minimum distances between the obstacle and manipulator arms, which are plotted in Fig. 4. From the subfigure (c), when Link 3 enters the outer safety threshold, that is, the distance to the obstacle is smaller than 10cm, the obstacle avoidance scheme repels it away. Fig. 4 also plots the distance between the obstacle and each link without the obstacle avoidance scheme. It is seen that in this case, Link 3 will penetrate into the obstacle (corresponding to zero distance). Figs. 5 and 6 depict the tracking velocity error and position error, both of which are very small. Fig. 7 shows the comparison of the minimized objective function with and without the obstacle avoidance scheme. It is reasonable to observe that the objective function value in the former case is greater than that in the latter case.

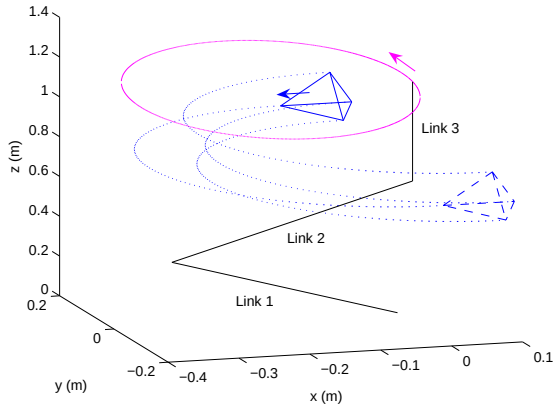


Fig. 2. Trajectories of the end-effector and the moving obstacle.

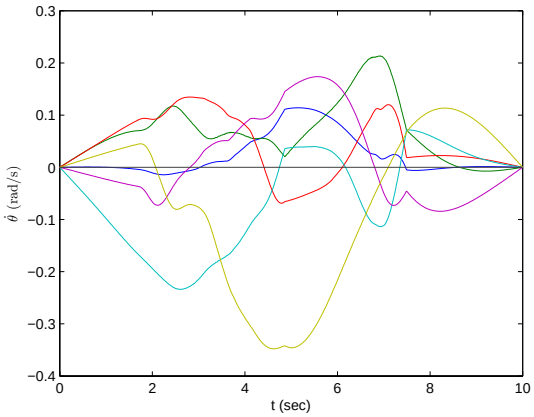


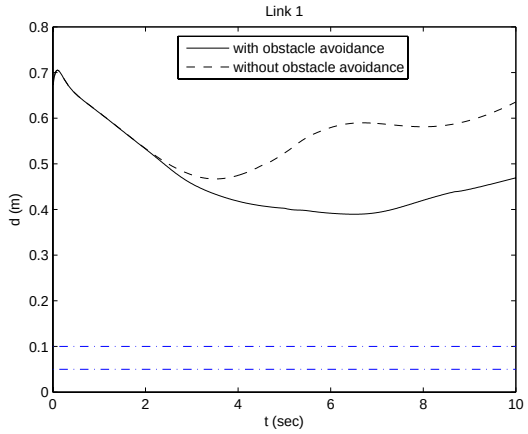
Fig. 3. Joint velocities.

V. Concluding Remarks

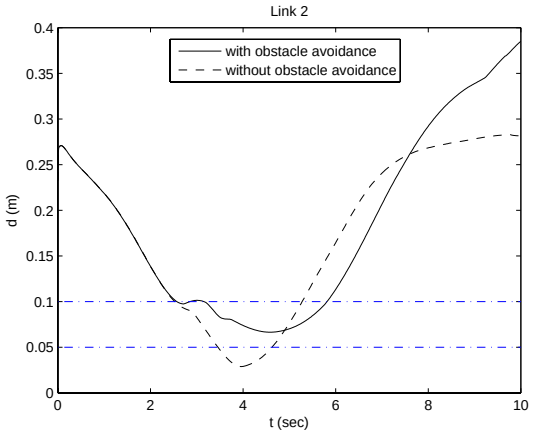
In some recent obstacle avoidance control schemes for kinematically redundant robot manipulators, the positional information of the the critical points on the manipulator related to the obstacles were assumed to be available *a priori*. In the paper, we provide a convex optimization problem formulation for determining such points by assuming convex-shape obstacles and serial-link manipulator. In particular, if the obstacle is a polyhedron, the problem becomes a quadratic programming (QP) problem. For solving this problem in real-time a general projection neural network (GPNN) is proposed.

A velocity smoothing technique is introduced into a previously established QP formulation of velocity control. Then, a simple neural network called the improved dual neural network (IDNN) is adopted to solve the problem in real-time.

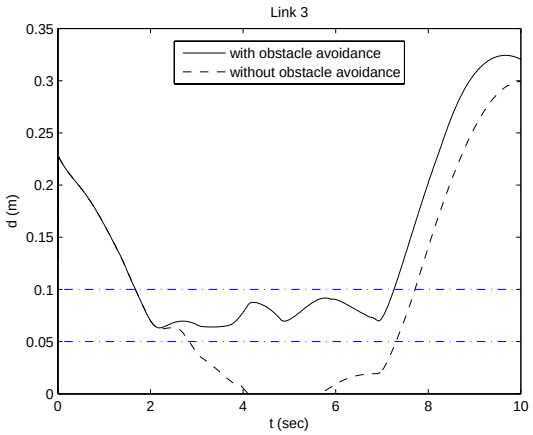
Compared with existing counterparts for solving the two QP problems using neural networks, the selected IDNN and GPNN possess lower model complexity and high perfor-



(a)



(b)



(c)

Fig. 4. Minimum distance between the obstacle and (a) Link 1, (b) Link 2, (c) Link 3.

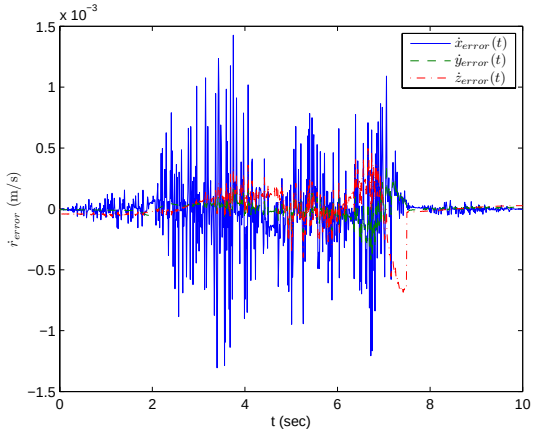


Fig. 5. Velocity error of the end-effector.

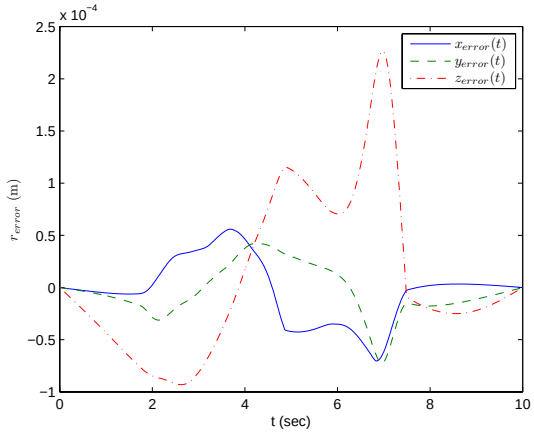


Fig. 6. Position error of the end-effector.

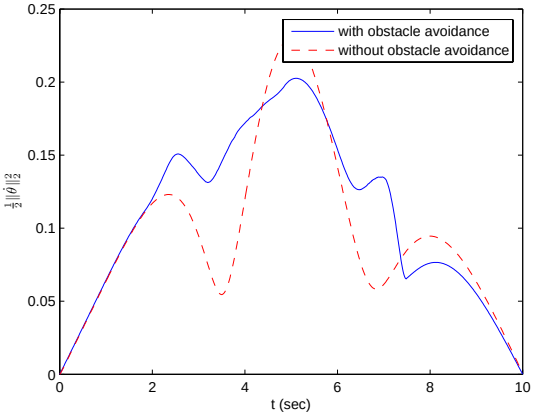


Fig. 7. Minimum of the objective function.

mance. The solutions to the time-varying QP problems using the neural networks are simulated based on the PA10-7C robot manipulator, which validate the effectiveness of the proposed approach.

REFERENCES

[1] B. Siciliano and O. Khatib, Eds., *Handbook of Robotics*. London: Springer, 2007.

[2] Y. Zhang and J. Wang, "Obstacle avoidance for kinematically redundant manipulators using a dual neural network," *IEEE Trans. Syst., Man, Cybern. B*, vol. 34, no. 1, pp. 752–759, Feb. 2004.

[3] S. Liu, X. Hu, and J. Wang, "Obstacle avoidance for kinematically redundant manipulators based on an improved problem formulation and the simplified dual neural network," in *Proc. IEEE Three-Rivers Workshop on Soft Computing in Industrial Applications*, Passau, Bavaria, Germany, Aug. 2007, pp. 67–72.

[4] I. Ohya, A. Kosaka, and A. Ka, "Vision-based navigation by a mobile robot with obstacle avoidance using single-camera vision and ultrasonic sensing," *IEEE Trans. Robot. Autom.*, vol. 14, no. 6, pp. 969–978, Dec. 1998.

[5] S. Shoval and J. Borenstein, "Using coded signals to benefit from ultrasonic sensor crosstalk in mobile robot obstacle avoidance," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, vol. 3, Seoul, Korea, May 2001, pp. 2879–2884.

[6] F. T. Cheng, T. H. Chen, Y. S. Wang, and Y. Y. Sun, "Obstacle avoidance for redundant manipulators using the compact QP method," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, vol. 3, Atlanta, Georgia, USA, May 1993, pp. 262–269.

[7] F. T. Cheng, Y. T. Lu, and Y. Y. Sun, "Window-shaped obstacle avoidance for a redundant manipulator," *IEEE Trans. Syst., Man, Cybern. B*, vol. 28, no. 6, pp. 806–815, Dec. 1998.

[8] H. Ding and S. K. Tso, "Redundancy resolution of robotic manipulators with neural computation," *IEEE Trans. Industrial Electronics*, vol. 46, no. 1, pp. 230–233, Feb. 1999.

[9] J. Wang, Q. Hu, and D. Jiang, "A lagrangian network for kinematic control of redundant robot manipulators," *IEEE Trans. Neural Netw.*, vol. 10, no. 5, pp. 1123–1132, Sep. 1999.

[10] Y. Xia, G. Feng, and J. Wang, "A primal-dual neural network for on-line resolving constrained kinematic redundancy in robot motion control," *IEEE Trans. Syst., Man, Cybern. B*, vol. 35, no. 1, pp. 54–64, 2005.

[11] Y. Zhang, S. S. Ge, and T. H. Lee, "A unified quadratic programming based dynamical system approach to joint torque optimization of physically constrained redundant manipulators," *IEEE Trans. Syst., Man, Cybern. B*, vol. 34, no. 5, pp. 2126–2132, Oct. 2004.

[12] E. A. Al-Gallaf, "Multi-fingered robot hand optimal task force distribution - neural inverse kinematics approach," *Robot. Auton. Syst.*, vol. 54, no. 1, pp. 34–51, Jan. 2006.

[13] S. Liu and J. Wang, "A simplified dual neural network for quadratic programming with its KWTA application," *IEEE Trans. Neural Netw.*, vol. 17, no. 6, pp. 1500–1510, 2006.

[14] X. Hu and J. Wang, "Solving generally constrained generalized linear variational inequalities using the general projection neural networks," *IEEE Trans. Neural Netw.*, vol. 18, no. 6, pp. 1697–1708, Nov. 2007.

[15] Y. Xia and J. Wang, "A general projection neural network for solving monotone variational inequalities and related optimization problems," *IEEE Trans. Neural Netw.*, vol. 15, no. 2, pp. 318–328, March 2004.

[16] B. Allotta, V. Colla, and G. Bioli, "Kinematic control of robots with joint constraints," *J. Dyn. Syst. Meas. Control-Trans. ASME*, vol. 121, no. 3, pp. 433–442, Sep. 1999.

[17] H. Ding and J. Wang, "Recurrent neural networks for minimum infinity-norm kinematic control of redundant manipulators," *IEEE Trans. Syst., Man, Cybern. A*, vol. 29, no. 3, pp. 269–276, May 1999.

[18] X. Hu and J. Wang, "Design of general projection neural networks for solving monotone linear variational inequalities and linear and quadratic optimization problems," *IEEE Trans. Syst., Man, Cybern. B*, vol. 37, no. 5, pp. 1414–1421, Oct. 2007.

[19] X. B. Gao, "A neural network for a class of extended linear variational inequalities," *Chinese Jounral of Electronics*, vol. 10, no. 4, pp. 471–475, Oct. 2001.

[20] X. Hu and J. Wang, "An improved dual neural network for solving a class of quadratic programming problems and its k-winners-take-all application," *IEEE Trans. Neural Netw.*, vol. 19, no. 12, pp. 2022–2031, Dec. 2008.