Obstacle Avoidance for Kinematically Redundant Manipulators Based on an Improved Problem Formulation and the Simplified Dual Neural Network

Shubao Liu, Xiaolin Hu and Jun Wang

Abstract—With the wide deployment of kinematically redundant manipulators in complex working environments, obstacle avoidance emerges as an important issue to be addressed in robot motion planning. In this paper, the inverse kinematic control of redundant manipulators with obstacle avoidance task is formulated into a (convex) quadratic programming (QP) problem with both equality and inequality constraints. Compared with our previous formulation, the new scheme is more favorable in the sense that it can yield better solutions to the control problem. To solve this time-varying QP problem in real time, a newly emergent recurrent neural network, simplified dual neural network, is adopted, which has lower structural complexity compared with existing neural networks for solving QP problems. The effectiveness of the proposed approach is demonstrated by using simulations on the Mitsubishi PA10-7C manipulator.

I. INTRODUCTION

As the development of automation industry, robot manipulators are required to work in more and more complex and dynamic environments. An important issue to be considered is how to effectively avoid static or moving objects in the workspace. Kinematically redundant manipulators are those having more degrees of freedom than required to perform given end-effector moving tasks. Being dexterous and flexible, they have been used for avoiding obstacles, singularity, and optimizing various performance criteria in addition to tracking desired end-effector trajectories. Of those versatile applications, obstacle avoidance is extremely important for successful motion control when obstacles exist in the workspace.

Many studies have been reported on using kinematically redundant manipulators for motion control and obstacle avoidance. The most popular method for real-time obstacle avoidance is to apply the pseudoinverse formulation for obtaining a general solution at velocity level, which contains a minimum infinity-norm (instead of Euclidean norm) of joint velocity vector or exact joint torques, many researchers formulate the inverse kinematic control problem explicitly into optimization problems [2], [9]–[14]. In particular, examples for obstacle avoidance include efforts in [2], [9], [10], [14]. Following these ideas, in this paper, we will formulate the inverse kinematics problem with obstacle avoidance into a quadratic programming (QP) problem, which is more effective compared with counterparts in the literature.

A main disadvantage of the optimization formulation based techniques is that they entail intensive computation at each step of the manipulator movement. For solving such time-varying problems the numerical algorithms are usually incompetent, especially in dynamic and/or uncertain environments. Parallel and distributed approaches such as recurrent neural networks to real-time obstacle avoidance have been widely explored since the work of Tank and Hopfield [15]. Reported results of numerous investigations have shown many advantages over traditional optimization algorithms (e.g., see [16], [17] and references therein). In particular, many neural networks have been developed to solve the inverse kinematics problems focusing on achieving pseudoinverse solutions [18], minimum torque [13], [19], optimal grasping force [20], and so on. In [14], a dual neural network is developed for minimizing the $L_2$-norm of joint velocity vector in presence of obstacles in workspace. Same problem is considered in the paper but with a different formulation and a new neural network possessing lower hardware complexity which is favored in circuit realization.

The remainder of this paper is organized as follows. In Section II, the obstacle avoidance problem is formulated into a time-varying QP problem. In Section III, a neural network for solving the QP problem is presented. Section IV reports the simulation results and Section V concludes the paper.

II. OBSTACLE AVOIDANCE SCHEMES

A. Equality-Constrained Formulation

The obstacle avoidance task is to identify from time to time the critical points, defined as the points on the manipulator...
link which are closest to the obstacles, and then assign them desired velocities which direct the critical point away from the obstacle. Each critical point \( C \) is defined as a point on the link \( L \) which is closest to the obstacle point \( O \). As shown in Fig. 1, corresponding to different relative positions of the obstacle and the link, there are two possible cases for locating the critical point \( C \) on the vulnerable link \( L \). Here the obstacle point \( O \) is the representative of the obstacle object. In model-based control, the position of the obstacle \( O \) is available \emph{a priori}, and is thus derived via the online distance minimization between the manipulator link and the obstacle [1], [4], [10], [21]; while in sensor-based control, it is determined by synthetic information of sensor fusion technology, e.g., utilizing vision, ultrasonic and infra-red sensors [22], [23].

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

\[
\begin{align*}
    r_e(t) &= f_e(\theta(t)), \\
    r_c(t) &= f_c(\theta(t))
\end{align*}
\]

(1)

where \( r_e \in \mathbb{R}^m \) is \( m \)-dimensional position and/or orientation vector of the end-effector, and \( r_c \in \mathbb{R}^3 \) is the position vector of critical point, \( \theta(t) \) is the joint vector of the manipulator, and \( f_e \) and \( f_c \) are nonlinear functions of the manipulator with respect to respectively the end-effector and the critical point. If there exist more than one critical points, multiple equations, similar to the second one, are 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 (1). 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

\[
J_e(\theta)\dot{\theta} = \hat{r}_e, \quad J_c(\theta)\dot{\theta} = \hat{r}_c,
\]

where \( J_e(\theta) = \partial f_e(\theta)/\partial \theta \) and \( J_c(\theta) = \partial f_c(\theta)/\partial \theta \) are the Jacobian matrices, and \( \hat{r}_e \) is the desired velocity of the end-effector, \( \hat{r}_c \) is the desired velocity of the critical point which should be properly selected to effectively direct the link away from the obstacle. Then the motion planning and obstacle avoidance problem of a kinematically redundant manipulator can be formulated in a time-varying local optimization framework as [1], [2], [5]:

\[
\begin{align*}
    \text{minimize} & \quad c(\dot{\theta}(t)) \\
    \text{subject to} & \quad J_e(\theta(t))\dot{\theta}(t) = \hat{r}_e(t) \\
                      & \quad J_c(\theta(t))\dot{\theta}(t) = \hat{r}_c(t) \\
                      & \quad l \leq \dot{\theta}(t) \leq u
\end{align*}
\]

(2)

where \( l \) and \( u \) are respectively lower and upper bounds of the joint velocity vector, and \( c(\cdot) \) is an appropriate convex cost function, for example, \( \|\dot{\theta}(t)\|_1, \|\dot{\theta}(t)\|_2, \|\dot{\theta}(t)\|_\infty \), selected according to the task needs. To avoid singularity configurations or drift in repeated operations, a linear term of \( \dot{\theta} \) can be added. The first constraint corresponds to the primary goal of tracking a given end-effector motion, and the second constraint is to achieve the secondary goal of obstacle avoidance, which is included whenever the shortest distance between the obstacle and the manipulator is within a prescribed safety margin. When there exist more than one obstacle, multiple constraints, similar to the second constraint in (2) should be included. By putting the motion requirement of the critical points as a constraint instead of a part of the objective function (e.g., [9]), the solution obtained from (2) is ensured to drive the manipulator to avoid obstacles compulsively, if only there exists a feasible solution to the optimization problem (2).

### B. Inequality-Constrained Formulation

The equality constrained formulation (2) has several drawbacks. First, how to determine the suitable magnitude of escape velocity \( \hat{r}_c \)? This value should be selected large enough to ensure that the link moves faster than the obstacle (if the obstacle is moving) to ensure the minimum distance between them enlarge as time goes on. At the same time, to be safe and energy efficient, the velocity should not be too large. Second, suppose that there are \( p \) critical points. If \( m + 3p > n \), the optimization problem (2) is overdetermined, i.e., it has no solution. In this sense, the equality constraints unnecessarily reduce the feasible solution space, sometimes even make it null. One possible approach to overcome these drawbacks is to replace the equality constraints by inequality constraints. As shown in Fig. 2, we can just constrain the direction and magnitude of \( \hat{r}_c \) in a range and let the optimization process to determine its accurate direction and magnitude.

The constraints of direction and magnitude of \( \hat{r}_c \) can be considered as follows. The component of \( \hat{r}_c \) projected on \( OC \) (i.e., \( CE \) in Fig. 2) should change with the minimum distance \( |OC| \) between the obstacle and the link. When the distance is large, \( |CE| \) can be small, but when the distance is near small, especially less than the safety margin, \( |CE| \) should be big enough to direct the link away from the obstacle promptly. Because the moving velocity of the obstacle is not known, \( |CE| \) should be large enough when the distance \( |OE| \) is very small. A natural candidate function is the hyperbolic
function \( \frac{c}{|OC|} \), where \( c \) is a positive parameter controlling the velocity magnitude. Another reason for choosing this candidate function is that the resulting inequality can be easily expressed in terms of \( \dot{\theta} \), which can be seen clearly in the following derivation. With this candidate function, the obstacle avoidance constraint can be formulated as

\[
|\dot{r}_c| \cos \alpha \geq \frac{c}{|OC|},
\]

or

\[
\dot{r}_c \cdot \hat{OC} \geq c. \tag{3}
\]

Because

\[
\hat{OC} = [x_c - x_o \ y_c - y_o \ z_c - z_o]^T,
\]

and

\[
\dot{r}_c = J_e(\theta) \dot{\theta},
\]

(3) can be written as

\[
[ x_c - x_o \ y_c - y_o \ z_c - z_o ] J_e(\theta) \dot{\theta} \geq c. \tag{4}
\]

If there are \( p \) critical points, define

\[
L(\theta) = \begin{bmatrix}
J_{c1}(\theta) \\
J_{c2}(\theta) \\
\vdots \\
J_{cp}(\theta)
\end{bmatrix}.
\]

Substituting the equality constraint in (2) by the inequality constraint (4), we get the following formulation with the \( L_2 \)-norm as the cost function,

\[
\begin{align}
\text{minimize} & \quad \frac{1}{2} \|\dot{\theta}(t)\|_2^2 \\
\text{subject to} & \quad J_e(\theta(t)) \dot{\theta}(t) = \dot{r}_c(t), \\
& \quad L(\theta(t)) \dot{\theta}(t) \geq c, \\
& \quad l \leq \theta(t) \leq u.
\end{align} \tag{5}
\]

If we fix time \( t \) in (5), it becomes a convex quadratic programming problem and can be solved by any optimization technique. But because that the parameters in (5) such as \( J_e \) and \( L \) are time-varying, such a solving procedure are required to repeat at each time instant. The entire kinematic control process is shown in Fig. 3, similar to most optimization based kinematic controls in [2], [9]–[14]. The desired end-effector velocity \( \dot{r}_e \) and the obstacle information, i.e., the position of the critical point \( C \) in Fig. 1 are fed into the optimizer as its inputs. In sensor based control, the critical point is detected with respect to the current pose of manipulator. The time-varying parameters of problem (5) 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 optimization scheme output. By further taking integration of the joint velocities with the known initial values, we can get the joint motions that the manipulator should follow.

\[ \dot{r}_e, \dot{\theta}, \theta \]

\[ \text{Manipulator} \]

\[ \text{Optimizor} \]

\[ \text{Sensors} \]

\[ \text{Obstacle detection} \]

\[ \text{C. Model Comparison} \]

It should be noted that a similar QP problem to (5) is formulated for the inverse kinematics problem with obstacle avoidance in [14]. That problem differs from (5) only in the inequality which represents the obstacle avoidance requirement. That’s to say, the inequality \( L(\theta) \dot{\theta} \geq c \) in (5) is replaced by

\[
J_N(\theta) \dot{\theta} \leq 0, \tag{6}
\]

where \( J_N(\theta) = -\text{sgn}(\hat{OC}) \circ J_C(\theta) \). The vector matrix multiplication operator \( \circ \) is defined as \( u \circ V = [u_1 V_1, u_2 V_2, \ldots, u_p V_p]^T \), where the column vector \( u = [u_1, u_2, \ldots, u_p]^T \) and the row vector \( V_i \) denotes the \( i \)th row of matrix \( V \). The geometric interpretation of (6) is that each component of \( \dot{r}_c \) should take the same sign as corresponding component of \( \hat{OC} \). In other words, the permissible directions of \( \dot{r}_c \) constitute 1/8 of the whole space \( R^3 \) in that \( \dot{r}_c \in R^3 \). However, from (3), the permissible directions of \( \dot{r}_c \) constitute one half of the whole space \( R^3 \) in the proposed scheme of this paper. In this sense, the new scheme enlarges the feasible solution space of the QP problem compared to that in [14]. A simulation study in Section IV will validate the superiority of the new scheme over that in [14]. On the other hand, from (6), each critical point results in three inequalities because \( J_N(\theta) \in R^{3 \times n} \); while in the new formulation (5), each critical point introduces only one inequality. It is well-known in the optimization context that adding constraints to the problem implies adding complexity to the problem, and in the neural network context, implies adding complexity to the neural network architecture.

As indicated above, compared to the equality formulations in [1], [2], [5] and the inequality formulation in [14], mathematically, the proposed inequality constraint in (5) has weakened the stringent requirements imposed by them on the feasible region of the QP problem to some extend. Nevertheless, it still does not assure that there always exists a solution to the problem. For example, too many obstacles in the workspace may result in an empty set of the feasible region of problem (5). By considering that such cases are encountered only in bad posed situations, where it is physically impossible to accomplish the motion task, for instance, because of the physical limits of the manipulator and environment, we hereafter assume there exists at least one solution to problem (5). Another weak assumption on the problem is that the rank of \( J_e \in R^{n \times n} \) is \( m \), since otherwise for a well posed
manipulator, several equalities in $J_e\dot{\theta} = \dot{r}_e$ can be removed by certain simple technique.

III. NEURAL NETWORK MODEL

The bottleneck of the proposed method in the last section for kinematic control of manipulators is the intensive computation for solving the QP problem (5) (see the block “Optimizer” in Fig. 3). If conventional numerical algorithms are adopted, a processing unit with very high performance is required; otherwise the motion of the manipulator will get slow. We here suggest a neural network for solving (5) specifically, which can be realized by a hardware system connected by a dense interconnection of simple analog computational elements. Because of the analog nature of the circuit system, neural networks have great potential in handling on-line optimization problems [15].

Rewrite (5) as follows

$$\begin{align*}
\text{minimize} & \quad \frac{1}{2}\|\dot{\theta}\|^2 \\
\text{subject to} & \quad J_e(\theta)\dot{\theta} = \dot{r}_e, \\
& \quad \xi^- \leq F(\theta)\dot{\theta} \leq \xi^+.
\end{align*}$$

where

$$F(\theta) = \begin{bmatrix} L(\theta) \\ \frac{\partial}{\partial \theta} \end{bmatrix}, \xi^- = \begin{bmatrix} c \\ l \end{bmatrix}, \xi^+ = \begin{bmatrix} \infty \\ u \end{bmatrix}.$$  

We adopt the simplified dual neural network invented in [24] to solve the above problem:

- state equation

$$\epsilon \frac{du}{dt} = -FMF^Tu + g(FMF^Tu - u + Fs) - Fs; \quad (8a)$$

- output equation

$$v = MF^Tu + s, \quad (8b)$$

where $M = I - J_e^T(J_e J_e^T)^{-1}J_e, s = J_e^T(J_e J_e^T)^{-1}r_e, u \in R^{n+p}$ is the state vector, $v = \dot{\theta}$ is the output vector, $\epsilon > 0$ is a scaling parameter that controls the convergence rate of the neural network, and $g(v)$ is a piecewise linear function defined as $g(v) = [g_1(v_1), g_2(v_2), \ldots, g_{n+p}(v_{n+p})]^T$, and for $i = 1, \ldots, n + p$,

$$g_i(v_i) = \begin{cases} 
\xi^-_i, & \text{if } v_i < \xi^-_i, \\
\xi^+_i, & \text{if } \xi^-_i \leq v_i \leq \xi^+_i, \\
\xi^+_i & \text{if } v_i > \xi^+_i.
\end{cases}$$

Because the analytic expression of $J_e, F$ can be obtained in the design stage, the analytic expressions of $M$ and $s$ can be computed beforehand, which will facilitate the circuits realization of the neural network. The architecture of the neural network is illustrated in Fig. 4 (for clarity, only the state equation part is shown). In the figure, the left part is about the state equation, where $\lambda = 1/\epsilon, F = \{w_i\}, F_s = \{u_i\}, i = 1, \ldots, N, j = 1, \ldots, N$, and $N = n + p$. The right part is about the output equation, where $MF = \{s_{ij}\}, s = \{s_i\}, v = \{v_i\}, i = 1, \ldots, n, j = 1, \ldots, N$. It is seen from the figure that only $n + p$ integrators are needed.

Neural network (8) is simpler in structure than its predecessor, the dual neural network in [14]. From [24], the state equation (8a) is stable in the Lyapunov sense and globally convergent to an equilibrium point, and the output $v$ of the neural network defined in (8b) globally convergent to an exact solution of problem (7). Moreover, the convergence rate increases when $\epsilon$ decreases.

IV. SIMULATION RESULTS

In this section, the validity of the proposed obstacle avoidance scheme and the real-time solution capability of the proposed neural network SDNN is 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 [13]. In this study, only the positioning of the end-point is concerned, then $m = 3$ and $n = 7$. The parameter $c$ is chosen as $5\epsilon - 4$. The parameter that controls the convergence rate of the neural network is chosen as $\epsilon = 10^{-6}$.

In this example, the desired motion of the end-effector is a circle of radius $r = 20cm$ with the revolute angle about the $x$ axis $\pi/6$. The task time of the motion is 10s and the joint variables are initialized as $\theta(0)=[0; -\pi/4; 0; \pi/2; 0; -\pi/4; 0]$. In the workspace, suppose that there are two obstacle points, [-0.0454m, -0.0737m, 0.8367m] and [-0.1541m, -0.0609m, 0.5145m]. (These obstacle points are denoted as black dot points in the following visualization figures.) Fig. 5 illustrates the simulated motion of the PA10 manipulator in the 3D workspace, which is sufficiently close to the desired one with the tracking error less than 0.6mm, as shown in Fig. 6.

Fig. 7 shows the comparison between the case with and without the obstacle avoidance scheme. Without obstacle avoidance, the link will collide with an obstacle as shown by the dotted line in Fig. 7. But when the proposed scheme is adopted, the link will move away from the obstacle once entering the danger zone, which is shown by the solid line in Fig. 7.

The output of the neural network $\dot{\theta}$ (i.e., control signal) is shown in Fig. 8. From this figure, we can see that the control signals change rapidly when the manipulator links are near the obstacles, but more smoothly than the scheme in [14]. Fig. 9 shows the comparison of the minimized objective function between the present scheme and the scheme in [14]. The optimization performance of the present scheme (shown by the solid line) is superior to that in the scheme in [14] (shown by the dotted line) with no abrupt performance exacerbation and better performance in most of the operation period. This fact responds the argument stated in the end of Section II-B about the sizes of feasible solution spaces of the two schemes.

V. CONCLUDING REMARKS

A new obstacle avoidance scheme for kinematically redundant manipulators control is proposed based on quadratic programming (QP) formulation. Compared with the existing QP formulations in the literature, the new scheme has larger feasible solution space and can thus yield better solutions in terms of optimizing objective functions. An efficient neural network is proposed to solve the problem in real-time. Compared with its existing counterparts for solving QP problems...
in neural network community, the proposed neural network possess lower hardware complexity but high performance. The solution to the time-varying QP problem using this neural network is simulated based on the PA10 robot manipulators, which validates the effectiveness of the proposed approach.

REFERENCES
Fig. 5. The PA10 manipulator in a circular motion

Fig. 6. Position Error of the end-effector

Fig. 7. Comparison of minimum distance between obstacle 2 and link 2 with and without obstacle avoidance

Fig. 8. Joint velocities

Fig. 9. Comparison of minimization of the objective function between two schemes


