On-Line Neural Network Algorithm for the Constrained Motion Planning of Redundant Manipulators

A. Ramdane-Cherif, D. Y. Meddah, V. Perdereau and M. Drouin
Lab. PARC, Univ. P. & M. Curie, 4, place Jussieu,
boite 164, 75252 Paris Cedex 05, France
e-mail: cherif@robo.jussieu.fr

Abstract

In this paper, we propose an iterative method using a neural network to solve the inverse kinematic problem for redundant manipulators in presence of motion constraints such as joint limits or obstacles. A constrained optimization scheme with penalty functions based on neural network is formulated. The neural network is adapted in the direction of decreasing a Lyapunov function to move the end-effector to the desired position while avoiding a collision with respectively a workspace object and a contact environment surface. This approach offers substantially better accuracy and avoids the computation of the inverse or pseudoinverse Jacobian matrix. The application of this scheme to a 3 degrees of freedom redundant manipulator is demonstrated through simulation results.

1 Introduction

A lot of researches have been carried out in the area of redundant robots, since these robots offer several advantages in dexterous motion tasks. Many authors have used the extra degrees of freedom of the redundant robots to optimize additional criteria when the given path in the workspace is tracked. Such criteria are the performance index that allows to satisfy: avoiding obstacles, keeping the joint coordinates within their limits, avoiding singularities, and improving dexterity [1] [2].

The most applications to obstacle and joint limit avoidance [3][4] have used one of the two main techniques for resolution of underspecified systems of equations: constrained generalized inverse-based approaches or augmented task space methods. Pin in [5] introduced a new method (Full Space Parameterization (FSP)) for the resolution of underspecified systems of algebraic equations. Then, he applied the FSP method to the constrained inverse kinematic problem. However, all these methods need a complicated formulation and a parameterized expression for the entire space of solutions for the basic system.

Our approach is based upon an optimization scheme using a neural network. The neural network is adapted in the direction of decreasing a Lyapunov function to move the end effector to the desired position. This approach exploits the redundancy to achieve some objective functions, and to satisfy some inequality constraints while tracking the desired end-effector trajectory. It does not require to compute the inverse or pseudoinverse Jacobian matrix. This method provides an accurate solution with only a few iterations per input point.

The organization of this paper is as follows. In section 2, we recall the kinematic formulation of robot manipulators. Section 3 is devoted to presenting the algorithm that we use to solve the kinematic problem for redundant and non-redundant manipulators. Simulation results of a three DOF robot arm are given in section 4. Finally, some conclusions are drawn in section 5.

2 Kinematic formulations

Let q be a $n \times 1$ vector of joint angles and x a $m \times 1$ vector of the corresponding Cartesian coordinates of the end-effector position ($n > m$), $r = n - m$ being the degree of redundancy. Then x and q are related by the forward kinematic transformation $f(\cdot)$ which is a well-known non linear function:

$$x = f(q) \tag{1}$$

One method to solve the inverse kinematic problem of redundant arms is to formulate it as an optimization problem with constraints, as follows:

$$\begin{cases} Minimize\ \Phi(q) \\ \text{subject to } x_d(t) - f(q) = 0 \text{ and } h(q) \leq 0 \end{cases} \tag{2}$$

where $\Phi(q)$ is a scalar kinematic objective function of the joint angles to be minimized, $x_d(t)$ is the desired end-effector trajectory and $h(q)$ is the inequality constraint vector.

For instance, when the redundancy is utilized to avoid collision with a workspace object, the distance between the object and the closest robot link $\beta(q)$ should exceed a certain threshold c, which leads to an inequality constraint of the form

$$h_i(q) = c_i - \beta_i(q) \leq 0, i = 1, \cdots, l \qquad (3)$$

where β_i is a kinematic function of the joint angles q and c_i is a constant. For each inequality in 3, two modes of operation are possible depending on q and c_i :

- Case One: $h_i(q) \leq 0$
  - In this case, the inequality constraint is satisfied and can be ignored. Therefore, the manipulator redundancy can be used to achieve the objective function $\Phi(q)$ while tracking the desired trajectory $x_d(t)$.
- Case Two: $h_i(q) > 0$
  - In this case, the inequality constraint is active, and the redundancy is utilized to satisfy the constraint and minimize an objective function $\Phi(q)$ while tracking the desired trajectory.

In our approach, we use the penalty-function methods to convert constrainced minimization like:

$$\left(\begin{array}{c} Minimize\ \Phi(q) \\ \text{subject to } h_i(q) \leq 0\ (i = 1, 2, \cdots, l) \\ q \in \Re^n \end{array} \right. \qquad (4)$$

into unconstrained optimization of an augmented objective function $\Omega(q)$:

$$\Omega(q) = \Phi(q) + \sum_{i=1}^{l} \alpha_i\ p\,(h_i(q)) \qquad (5)$$

where l is the number of constraints, α_i is the i^{th} penalty multiplier and p the penalty-function [6]. p may be an exterior penalty function $p(h_i) = h_i^2 \Gamma(h_i)$ ($\Gamma(h_i)$ is the Heaviside function) or an interior penalty function $p(h_i) = \frac{-1}{h_i}$

3 Principle of the proposed method

Instead of off-line training, our approach Fig. 1 trains on-line a neural network in order to approximate the inverse kinematic model by a linear function at each trajectory point [7]. For each point X_d, the optimal weight parameters are obtained via a Widrow-Hoff training algorithm as described below.

For this purpose, we introduce an extended position vector as:

$$X = F(q) = \left[\begin{array}{c} x \\ 0 \end{array} \right] = \left[\begin{array}{c} f(q) \\ g(q) \end{array} \right] \qquad (6)$$

•$f(q) \in \Re^m$ is the forward kinematic vector defined in (1).

•$g(q) \in \Re^r$ is the constraint vector added in the redundant case. These constraints can be represented by the general form proposed in Ballieul [8]:

$$g(q) = N^T \frac{\partial \Omega(q)}{\partial q} = 0 \qquad (7)$$

where N is the $n \times (n-m)$ null space matrix of J which corresponds to the self motion of a redundant arm:

$$N = \det(J_a) \left[\begin{array}{c} J_a^{-1} J_b \\ -I_{n-m} \end{array} \right] = \left[\begin{array}{c} AJ_b \\ -\det(J_a) I_{n-m} \end{array} \right] \qquad (8)$$

where $J_a^{-1} = A/\det(J_a)$, J_a is a m-square matrix made of the m first columns of J and J_b a $m \times (n-m)$ matrix of the remaining columns: $J = \left[\begin{array}{cc} J_a & J_b \end{array} \right]$, A is the cofactor matrix of J_a^T, and I_{n-m} is the $n-m$ identity matrix.

$\Omega(q)$ is the objective function associated with the redundancy problem:

•Without inequality constraints:

In this case, $\Omega(q) = \Phi(q)$ is simply the scalar kinematic objective function. We have for each component j:

$$\frac{\partial \Omega(q)}{\partial q_j} = \frac{\partial \Phi(q)}{\partial q_j}$$

•With inequality constraints:

From (5), we have respectively for each component j:

a) Exterior penalty method:

$$\frac{\partial \Omega(q)}{\partial q_j} = \frac{\partial \Phi(q)}{\partial q_j} + 2\alpha_j h_j(q) \frac{\partial h_j(q)}{\partial q_j} \Gamma(h_j)$$

b) Interior penalty method:

$$\frac{\partial \Omega(q)}{\partial q_j} = \frac{\partial \Phi(q)}{\partial q_j} + \alpha_j \frac{1}{h_j(q)^2} \frac{\partial h_j(q)}{\partial q_j}$$

The problem is to find the inverse solution of Eq.6, i.e. the vector q_d so that

$$F(q_d) = X_d = \left[\begin{array}{c} x_d \\ 0 \end{array} \right]$$

3.1 The based algorithm

For a given desired Cartesian position, the objective is to approximate the kinematic inverse model by a linear function at each trajectory point i.e. to find q which satisfies the forward mapping $x = f(q)$ while optimizing the given performance index $\Phi(q)$ and satisfying the additional constraints.

At each iteration c for k^{th} point, this algorithm involves two phases:

- During the first phase, the input X_k^d (the desired extended end-effector position) is presented and propagated through the network to compute the output value:

$$q_k^c = q_{d_{k-1}} + \tau\ W_k^c\ X_k^d \tag{9}$$

$$(q_k^c)_i = \begin{cases} 2\pi + \gamma & \text{if } (q_k^c)_i > 2\pi \\ (q_k^c)_i & \text{if } (q_k^c)_i \leq 2\pi \\ -2\pi + \gamma & \text{if } (q_k^c)_i < -2\pi \end{cases} \tag{10}$$

  where $(q_k^c)_i$ is the i^{th} component of q_k^c, γ is a random variable having a small variance to avoid the possible cyclic trajectory of the joint position, $q_{d_{k-1}}$ is the desired joint position obtained for the $(k-1)^{th}$ point which is considered as the bias of the neural network and τ is the sampling period. The initial value W_k^0 is the optimal weight matrix W_{k-1}^* obtained for the $(k-1)^{th}$ point.

  Then, the vector q_k^c is used to compute the end-effector position:

$$X_k^c = F(q_k^c) \tag{11}$$

  and the error is given by:

$$E_k^c = X_k^d - X_k^c = \begin{bmatrix} \varepsilon_k^c \\ e_k^c \end{bmatrix} \tag{12}$$

- During the second phase, the weights of the neural network is adjusted according to the delta rule [9]:

$$W_k^{c+1} = W_k^c + \eta \triangle W_k^c \tag{13}$$

  where η is the reduction factor $(0 < \eta < 2)$.

In our previous works [10], the search direction involved an inverse Jacobian matrix calculation. This solution is time consuming and problems may arise in the vicinity of singular points.

3.2 A new solution

We propose here a new method avoiding these drawbacks and leading to a very quick and efficient solution. The main idea is to use a Lyapunov function for calculating at each iteration the search direction $\triangle W$.

For a given initial weight matrix W, the algorithm updates W iteratively in the direction of decreasing the Lyapunov function defined in terms of the errors:

$$V = \frac{\lambda}{2}\varepsilon^T\varepsilon + \frac{1}{2}e^Te \tag{14}$$

where $\varepsilon = x^d - x = x^d - f(q)$, $e = 0 - g(q)$, x_d is the desired value of x and λ is the Lagrangian multiplier. The purpose of this multiplier is to force ε to converge to zero by increasing λ exponentially when q is near the solution of ε.

For sake of simplicity, we eliminate in the following equations the index k and c.

The time derivative of the Lyapunov function Eq.14 is given by:

$$\dot{V} = \left(\frac{\partial V}{\partial q}\right)^T \dot{q} + \frac{\partial V}{\partial \lambda}\dot{\lambda} \tag{15}$$

$$\dot{V} = -(\lambda J^T\varepsilon + J_e^Te)^T\ \dot{q} + \frac{1}{2}\|\varepsilon\|^2\ \dot{\lambda} \tag{16}$$

where $J = \frac{\partial f(q)}{\partial q}$ and $J_e = \frac{\partial g(q)}{\partial q}$, by differentiating Eq.9, we obtain $\dot{q} = \tau\ \dot{W}\ X^d$. For simulation on a digital computer, we use a discrete time update rule: $\dot{W} \simeq \frac{\triangle W}{\tau}$ and $\dot{\lambda} \simeq \frac{\triangle \lambda}{\tau}$. Then, Eq. 16 becomes:

$$\dot{V} = -\ (\lambda J^T\varepsilon + J_e^Te)^T\ \triangle W\ X^d + \frac{1}{2\tau}\|\varepsilon\|^2\triangle\lambda \tag{17}$$

If we set:

$$\dot{V} = -\frac{\lambda}{2}\|\varepsilon\|^2 \tag{18}$$

Then we can deduce the correction terms ΔW and $\Delta\lambda$:

$$\triangle W = \frac{\frac{\lambda}{2}\|\varepsilon\|^2 + \frac{1}{2\tau}\|\varepsilon\|\,\|e\|}{\|\lambda J^T\varepsilon + J_e^Te\|^2}(\lambda J^T\varepsilon + J_e^Te)\frac{SGN(X^d)^T}{(X^d)^T\ SGN(X^d} \tag{19}$$

and

$$\triangle\lambda = \frac{\|e\|}{\|\varepsilon\|} \tag{20}$$

where

$$SGN(X) = \begin{bmatrix} sgn(x_i) \\ \vdots \\ sgn(x_n) \end{bmatrix}$$

$$sgn(x_i) = \begin{cases} +1 \text{ if } x_i > 0 \\ -1 \text{ if } x_i < 0 \end{cases}$$

By substituting Eq.19 and Eq.20 in Eq.17 we obtain Eq.18. This result implies that $\dot{V} < 0\ \forall\varepsilon \neq 0$ and $\dot{V} = 0$ iff $\varepsilon = 0$.

The update of W based on $\triangle W$ determined by Eq.19 guarantees the convergence. The weights of the neural network and the Lagrangian multiplier are adjusted according to

$$W_k^{c+1} = W_k^c + \eta_1 \triangle W_k^c \quad (21)$$

$$\lambda^{c+1} = \lambda^c + \eta_2 \triangle \lambda^c \quad (22)$$

where η_i is the reduction factor $(0 < \eta_i < 2)$.

We must emphasize that no Jacobian inverse matrix calculation are necessary here. Furthermore, if the desired trajectory does not present large discontinuities, each new desired point X_k^d is very close of the previous one X_{k-1}^d and the error is small. This algorithm converges very quickly towards the solution q_d^k with few operations per iteration.

4 Simulation

A 3 d-o-f planar robot is considered to show the validity of our method. The forward kinematic function is:

$$f(q) = \begin{pmatrix} l_1c_1 + l_2c_{12} + l_3c_{123} \\ l_1s_1 + l_2s_{12} + l_3s_{123} \end{pmatrix} \quad (23)$$

and the $(n-m)$ null space vector of J is :

$$N = \begin{bmatrix} -l_2l_3s_3 & l_1l_3s_{23} + l_2l_3s_3 & -l_1l_2s_2 - l_1l_3s_{23} \end{bmatrix} \quad (24)$$

where $s_{ij} = \sin(q_i+q_j)$, $c_{ij} = \cos(q_i+q_j)$. Any convex objective function may be used in our approach. The parameters involved in the simulation are listed as $(\eta_1 = 1.5, \eta_2 = 0.01, \lambda(0) = 2)$.

For the two simulations that we present, the algorithm is stopped after a predefined number of iterations. For each trajectory point, this iteration number is fixed to eight iterations and is sufficient to find the solution. Good results have been obtained since the tracking error is very small and the criterion $g(q)$ is near the minimum for the whole trajectory.

The desired objective function is taken as: $\Phi(q) = \frac{1}{2}\sum_{i=1}^{3} l_i(q_{ik} - q_{i(k-1)})^2$. Where l_i is the length of the i^{th} link, $q_{i(k-1)}$ and q_{ik} are respectively the initial and current value of the i^{th} joint corresponding to the k^{th} point trajectory. This criterion tends to give the minimum joint displacement to move from one configuration to another.

4.1 The first simulation

For the first simulation the additional inequality constraint $h(q) = c - \beta(q) \leq 0$ ensure that the distance between the obstacle object and the closest robot link l_1 should exceed a certain threshold c. The obstacle is considered as a circle with a center $(x_b, y_b) = (0.4, 0.8)$ and a radius $c = 0.5m$ (Fig.2).

The solution must be such that the manipulator tracks the desired trajectory x_d, the objective function $\Phi(q)$ is minimized and the inequality constraints $h(q)$ are satisfied.

- Fig.3 represent the arm configurations for the objective function $\Phi(q)$ without inequality constraints. In this case we can not avoid the collision with the workspace obstacle.
- By applying inequality constraints, appropriate motion of the arm is now obtained, as show by Fig.4 representing the arm configuration using the interior penalty method ($\alpha = 0.01$). The results show that the end-effector converges successfully to the desired position while minimizing an objective function and avoiding a collision with a workspace object.

4.2 The second simulation

In this simulation, we consider the end-effector motion on the environment surface (Fig.5) represented here by the straight line:

$y - ax - b = 0$ with $a = -1$ and $b = 2$, (x, y) are the Cartesian coordinates in the base frame.

We decide to minimize the objective function $\Phi(q)$ defined above and ensure that the arm will not go over the contact surface. To this end, we maintain a constant distance β_0 between the last link position $(x_2, y_2) = (l_1c_1 + l_2c_{12}\ ,\ l_1s_1 + l_2s_{12})$ and the contact surface. The additional constraint is $h(q) = \beta(q) - \beta_0 \leq 0$ with $h(q) = y_2 - ax_2 - b$.

- Fig.6 exhibit the arm configuration for the objective function $\Phi(q)$ without inequality constraints. The end-effector coordinates track very closely the desired trajectory. However these configuration are not attainable in practice.
- By applying inequality constraints and using the interior penalty method ($\alpha = 0.01$), the arm configuration during the task is seen in Fig.7. While the end-effector moves, the objective function is well minimized and the last link position remains always inside the feasible region and far from the frontier of the allowable workspace.

5 Conclusion

In this paper, we have presented a new approach to solve the inverse kinematic problem of redundant manipulators. Our method is based on formulating a simple constrained optimization problem with

penalty methods using neural network. The neural network is adapted in the direction of decreasing the Lyapunov function to move the end effector to the desired position while minimizing an objective function and avoiding a collision with a workspace object or a contact surface environment. This method achieves an accurate solution with only a few iterations per input point and requires only the computation of the direct kinematic functions.

References

[1] D. T. K. Cleary, "Incorporating multiple criteria in the operation of redundant manipulators," in *IEEE International Conference on Robotics and Automation*, (Cincinnati, USA), pp. 618–624, 1990.

[2] H. Seraji and R. Colbaugh, "Improved configuration control for redundant robots," *Journal of Robotic Systems*, vol. 7, no. 6, pp. 897–928, 1990.

[3] P. Chiacchio, S. Chiaverini, L. Schiavicco, and B. Siciliano, "Closed-loop inverse kinematics schemes for constrained redundant manipulators with task space augmentation and task priority strategy," *International Journal of Robotics Research*, vol. 10, no. 4, pp. 410–425, 1991.

[4] R. Colbaugh, H. Seraji, and K. Glass, "Obstacle avoidance for redundant robots using configuration control," *Journal of Robotic Systems*, vol. 6, no. 6, pp. 721–744, 1989.

[5] F. G. Pin and F. A. Tulloch, "Resolving kinematic redundancy with constraints using the fsp (full space parameterization) approach," in *International Conference on Robotics and Automation*, (Minneapolis, Minnesota), pp. 468–473, April 1996.

[6] L. Scales, "Introduction to non-linear optimization," in *Springer-Verlag*, (New york), 1985.

[7] A. R-Cherif, V. Perdereau, and M. Drouin, "Inverse kinematic resolution using neural network," in *IASTED International Conference on Robotics and Manufacturing*, (Cancun, Mexico), pp. 170–172, June 1995.

[8] J. Baillieul, "Kinematic programming alternatives for redundant manipulators," in *IEEE International Conference on Robotics and Automation*, (St Louis), pp. 722–728, 1985.

[9] H. J. Sira-Ramirez and S. H. Zak, "The adaptation of perceptrons with applications to inverse dynamics systems," *IEEE Transaction on Systems, Man and Cybernetics*, vol. 21, no. 3, pp. 634–643, 1991.

[10] A. R.-Cherif, V. Perdereau, and M. Drouin, "Penalty approach for a constrained optimization to solve on-line the inverse kinematic problem of redundant manipulators," in *Proc. Of ICRA'97*, (Minneapolis, USA), Apr. 1996.

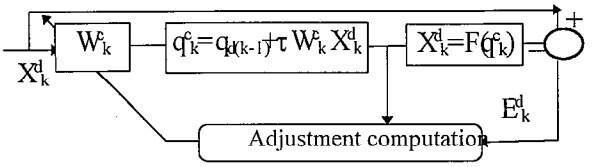

Figure 1: The proposed inverse kinematic scheme

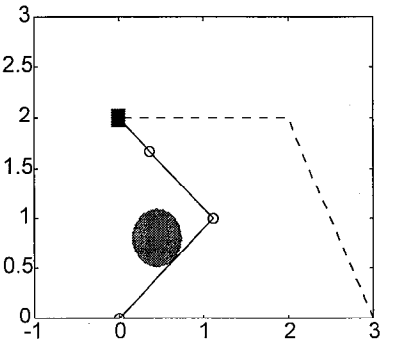

Figure 2: avoiding obstacle

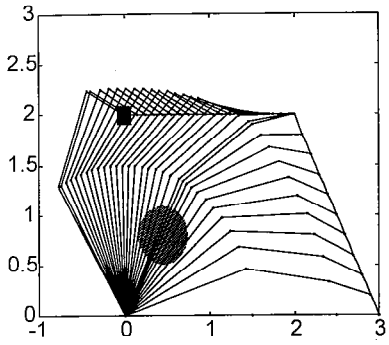

Figure 3: Arm configuration using $\Phi(q)$ without constraints

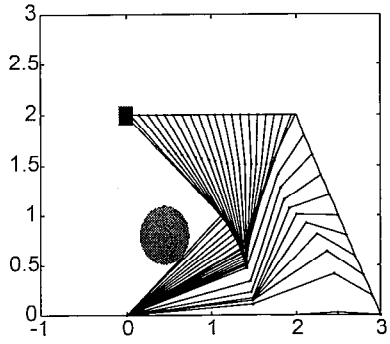

Figure 4: Arm configuration using $\Phi(q)$ with constraints

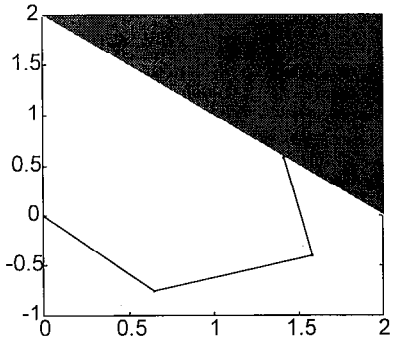

Figure 5: End-effector motion on the environment surface

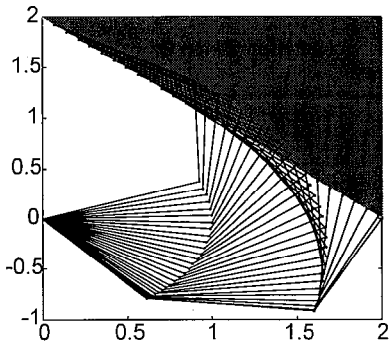

Figure 6: Arm configurations using $\Phi(q)$ without constraints

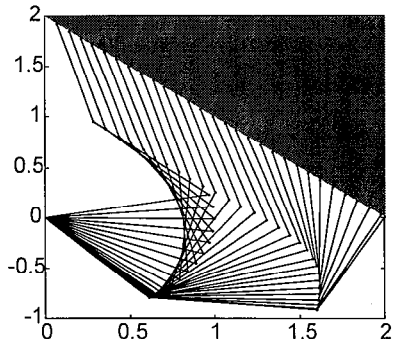

Figure 7: Arm configuration using $\Phi(q)$ with constraints