

Neural Network-Based Robot Trajectory Generation

Dan Simon *
TRW Systems Integration Group
PO Box 1310
San Bernardino, CA 92402

Abstract— Interpolation of minimum jerk robot joint trajectories through an arbitrary number of knots is realized using a hardwired neural network. Minimum jerk joint trajectories are desirable for their similarity to human joint movements and their amenability to accurate tracking. The resultant trajectories are numerical rather than analytic functions of time. This application formulates the interpolation problem as a constrained quadratic minimization problem over a continuous joint angle domain and a discrete time domain. Time is discretized according to the robot controller rate. The neuron outputs define the joint angles (one neuron for each discrete value of time) and the Lagrange multipliers (one neuron for each trajectory constraint). An annealing-type method is used to prevent the network from getting stuck in a local minimum. This paper discusses the optimizing neural network and its application to robot path planning, presents some simulation results, and compares the neural network method with other minimum jerk trajectory planning methods.

I. INTRODUCTION

This section first gives a brief review of robot path planning. Then an introduction to robot path planning using neural networks is presented.

A. Path Planning

The industrial robot is a highly nonlinear, coupled multi-variable system with nonlinear constraints. For this reason, robot control algorithms are often divided into two stages: *path planning* and *path tracking* [1]. Path planning is often done without much consideration for the robot dynamics, and with simplified constraints. This reduces the computational expense of the path planning algorithm.

*This work was performed while the author was with Syracuse University.

The output of the path planning algorithm is then input to a path tracking algorithm.

There are algorithms for the robot control problem which do not separate path planning and path tracking. These algorithms take source and destination Cartesian points as inputs, and determine optimal joint torques. The details of the robot dynamics and constraints are taken into account at various levels of complexity. A concise review of such algorithms is given in [10]. While such methods are attractive in that they provide optimal solutions to some robot control problem, they result in impractically complicated algorithms and a large computational expense. For this reason, several researchers [12, 15] have simplified the problem as follows: given a desired path in Cartesian space, derive the optimum joint trajectories subject to the full dynamics and constraints. While this approach reduces the computational expense somewhat, the expense is still much too high for on-line implementation. Some researchers [11, 13] assume not that the path is specified, but that it has some known parameterized form. They then optimize the path parameters with respect to some objective function.

A simpler approach to the robot control problem is to generate a suboptimal joint trajectory, and then track the trajectory with a controller. This approach ignores most of the dynamics of the robot. So the resultant trajectories do not take full advantage of the robot's capabilities, but are computationally much easier to obtain. In this approach, a number of knot points are chosen along the desired Cartesian path. The Cartesian knots are then mapped into joint knots using inverse kinematics. Finally, for each robot joint, an interpolating curve is fit to the joint knots. Some of the initial and final derivatives of the curve are constrained to zero so as to ensure that the robot begins and ends its motion smoothly. "Smoothness" is a concept which combines the ideas of derivative continuity and derivative magnitude. The interpolating curve provides the path tracker with joint angles and derivatives at the controller rate.

The most popular type of interpolation is algebraic splines [8, 16]. Higher order splines result in continuity of higher order derivatives, which reduces wear and tear on the robot [1], but this is at the expense of large oscillations of the trajectory. Trigonometric splines can be used to provide a less oscillatory interpolating curve [13].

B. Neural Networks Applied to Robot Path Planning

Neural networks have been applied to many fields of engineering, and the field of robotics is no exception [18, pp. 960-966]. This paper presents a new application of a hard-wired optimization network to the problem of trajectory generation through a set of given knots.

Consider a sequence of knots through which an interpolating curve is desired to pass. A human could create an interpolating curve, but in a different way than a computer algorithm would. Computer algorithms can calculate analytic functions which pass through given knots. A human can draw a smooth curve through a given set of knots, but without performing any mathematical calculations. In contrast with the computer algorithm, the interpolating curve drawn by the human would not be an analytic function of time. In addition, the human would not satisfy the constraints exactly, but only approximately. Such a result would be satisfactory for most robot path planning applications. These facts indicate that an artificial neural network may be able to do well at interpolation.

The robot path planning problem can be viewed as an optimization problem. Given a desired set of knots and endpoint constraints, find the "best" interpolating curve such that the knot errors and endpoint derivatives are not too "large." This problem is reduced in this paper to a constrained quadratic programming problem. Several researchers have solved quadratic optimization problems using neural networks [5, 19], but many of the networks make assumptions which limit the class of problems which can be solved. For instance, a norm of the weighting matrix of the objective function may be required to satisfy some inequality. Platt and Barr [9] formulate a neural network which can calculate a minimum of a general function subject to inequality or equality constraints. This is the network which is used in this paper to determine an optimal robot path through a given set of knots.

In order to plan an optimal robot trajectory, the measure of optimality must be defined. Human arm movements satisfy some optimality criterion, and this would seem to be a desirable criterion to adopt when planning trajectories for robot arms. Previous work [2, 3] suggests that human arm movements minimize a measure of Cartesian jerk or joint jerk. Others [6, 17] argue that the objective function is a measure of the derivative of the joint torques, and propose a neural network to learn such a trajectory. In this paper, a joint jerk objective function

is used. While this choice ignores the dynamics of the robot, it reduces the error of the path tracker [7] and thus is suitable for robotics applications.

Pontryagin's minimum principle has been used to analytically determine minimum joint jerk trajectories between two points subject to the constraints of zero velocity and acceleration at the endpoints [7]. The minimum Cartesian jerk trajectory between two points has also been derived [17]. But numerical methods must be used if more than two knots are given. This paper presents a method which is used to plan minimum joint jerk trajectories through an arbitrary number of knots.

II. CONSTRAINED NEURAL NETWORK-BASED MINIMIZATION

Platt and Barr [9] formulate a neural network which can be used for constrained minimization. Their algorithm, along with some straightforward extensions, is summarized in this section. Consider the following constrained minimization problem:

$$\min f(\vec{x}) \quad \text{subject to} \quad \vec{g}(\vec{x}) = 0 \quad (1)$$

where $f(\cdot)$ is a scalar functional, \vec{x} is an n -vector of independent variables, and $\vec{g}(\cdot)$ is a vector-valued function mapping $\mathcal{R}^n \rightarrow \mathcal{R}^m$.

Lagrange multipliers can be used to convert the constrained problem of (1) to the following unconstrained problem:

$$\min [f(\vec{x}) + \vec{\lambda}^T \vec{g}(\vec{x})] \quad (2)$$

where $\vec{\lambda}$ is an m -vector of Lagrange multipliers associated with the constraints $\vec{g}(\cdot)$. A necessary condition for the solution of (2) is

$$\frac{\partial f}{\partial \vec{x}} + \vec{\lambda}^T \frac{\partial \vec{g}}{\partial \vec{x}} = 0. \quad (3)$$

Now consider a neural network with dynamics of the form

$$\begin{aligned} \dot{x}_i &= -\frac{\partial f}{\partial x_i} - \sum_{\alpha=1}^m (\lambda_\alpha + c_\alpha g_\alpha) \frac{\partial g_\alpha}{\partial x_i} \\ &\quad (i = 1, \dots, n) \\ \dot{\lambda}_j &= c_j g_j \quad (j = 1, \dots, m) \end{aligned} \quad (4)$$

where \vec{c} is an m -vector of constants. Assume that the constraints $\vec{g}(\cdot)$ of the original problem (1) are linear functions of \vec{x} . Then differentiating \dot{x}_i in (4) gives

$$\begin{aligned} \ddot{x}_i + \sum_{j=1}^n \frac{\partial^2 f}{\partial x_i \partial x_j} \dot{x}_j + \\ \sum_{\alpha=1}^m \left[c_\alpha \left(g_\alpha + \sum_{j=1}^n \frac{\partial g_\alpha}{\partial x_j} \dot{x}_j \right) \frac{\partial g_\alpha}{\partial x_i} \right] = 0. \end{aligned} \quad (5)$$

Now consider the candidate Lyapunov energy function

$$E = \frac{1}{2} \sum_{i=1}^n (\dot{x}_i)^2 + \frac{1}{2} \sum_{\alpha=1}^m c_\alpha g_\alpha^2. \quad (6)$$

The derivative of this energy function is

$$\begin{aligned} \dot{E} &= \sum_{i=1}^n \left(\dot{x}_i \ddot{x}_i + \sum_{\alpha=1}^m c_\alpha g_\alpha \frac{\partial g_\alpha}{\partial x_i} \dot{x}_i \right) \\ &= - \sum_{i,j=1}^n \dot{x}_i \dot{x}_j \left(\frac{\partial^2 f}{\partial x_i \partial x_j} + \sum_{\alpha=1}^m c_\alpha \frac{\partial g_\alpha}{\partial x_j} \frac{\partial g_\alpha}{\partial x_i} \right) \\ &= -\dot{\vec{x}}^T A \dot{\vec{x}} \end{aligned} \quad (7)$$

where the element in the i^{th} row and j^{th} column of matrix A is given by

$$A_{i,j} = \frac{\partial^2 f}{\partial x_i \partial x_j} + \sum_{\alpha=1}^m c_\alpha \frac{\partial g_\alpha}{\partial x_j} \frac{\partial g_\alpha}{\partial x_i}. \quad (8)$$

It has been shown [9] that there exists a finite vector \vec{c} such that matrix A is positive definite at the constrained minima of (1). If A is continuous, then it is positive definite in some region surrounding each constrained minimum. Therefore if the dynamic system defined by (4) begins in that region and remains in that region, \dot{E} will remain less than zero unless $\dot{x}_i = 0 \forall i$. The energy E will therefore achieve a minimum when $\dot{x}_i = 0 \forall i$. But E is not at a minimum unless each of the constraints $\vec{g}(\vec{x})$ are zero, since $\vec{g}(\cdot)$ is linear. Therefore the system will settle into the zero-energy state where

$$\dot{\vec{x}} = 0 \quad (9)$$

$$\vec{g}(\vec{x}) = 0. \quad (10)$$

Now $\vec{g}(\vec{x}) = 0$ implies that the original constraints are satisfied, and $\dot{\vec{x}} = 0$ implies (4) that

$$\begin{aligned} \frac{\partial f}{\partial x_i} + \sum_{\alpha=1}^m (\lambda_\alpha + c_\alpha g_\alpha) \frac{\partial g_\alpha}{\partial x_i} &= \\ \frac{\partial f}{\partial \vec{x}} + \vec{\lambda}^T \frac{\partial \vec{g}}{\partial \vec{x}} &= 0 \end{aligned} \quad (11)$$

which satisfies the necessary conditions for a local minimum of the original constrained problem (see (3)). To sum up, (4), with an appropriately chosen \vec{c} , converges to a solution of the original constrained minimization problem of (1). Equation (4) is in the form of a first-order differential equation, which implies that it can be implemented in parallel hardware to yield a very quick solution.

III. THE SOLUTION OF THE PATH PLANNING PROBLEM

When interpolating the path of a robot joint between a set of joint space knots, it is desirable to obtain as smooth a solution as possible. This results in an appearance of coordination [2], reduces wear on the robot joints and prevents the excitation of resonances [1], and improves the accuracy of the path tracker [7]. Therefore, in robot trajectory generation, the interpolation problem for each joint can be stated as follows.

Given a set of L knots for a robot joint, determine a function $\theta(t)$ which

- is as "smooth" as possible;
- has "small" errors at the knots; and
- has "small" derivatives at the endpoints.

Smoothness can be defined as the integral of the square of the jerk of the position trajectory [2]. In order for the robot joint to start and stop its motion in a smooth manner, the first three derivatives at the endpoints should be small. If the path length is T seconds, and the desired knot angles are $\theta(t_j) = \phi_j$ ($j = 1, \dots, L$), then the optimization problem for each joint can be written as

$$\min \int_0^T [\theta'''(t)]^2 dt \quad (12)$$

$$\begin{aligned} \text{subject to } \theta(t_j) &= \phi_j & (j = 1, \dots, L) \\ \theta^{(r)}(0) = \theta^{(r)}(T) &= 0 & (r = 1, 2, 3) \end{aligned}$$

If the L knots are equally spaced in time, then the knot times t_i satisfy

$$t_i = (i-1)T/(L-1) \quad (i = 1, \dots, L). \quad (13)$$

The joint trajectory at the endpoints is exactly constrained. That is, the joint angles at $t = 0$ and $t = T$ are fixed constants. But the joint angles at the interior knot times are not truly equality constraints; the interior knot angles are more like centers of tolerance near which the joint trajectory is required to pass. Also, the first three endpoint derivatives do not need to be exactly zero. As long as they are very small, the robot motion will begin and end smoothly. Therefore, the constraints $\theta(t_1) = \phi_1$ and $\theta(t_L) = \phi_L$ can be considered "hard" constraints, while the remaining $(L+4)$ constraints in (12) can be considered "soft" constraints.

Since the joint trajectory is input to the path tracker at discrete values of time, the trajectory does not need to be a continuous function of time. It can be a discrete set of joint angles, defined only at times kh ($k = 0, 1, \dots, N$) where h is the sample period of the path tracker (typically

on the order of 0.01 seconds), and Nh is the length of the trajectory.

The angle θ_i is input to the path tracker every h seconds, starting at $t = 0$ and ending at $t = T$. There are exactly M discrete times per knot, so each knot angle is separated from its neighboring knots by Mh seconds. Thus the path length T satisfies

$$T = M(L - 1)h. \quad (14)$$

Also, from $t = 0$ to $t = T$, there are exactly $N + 1$ discrete time steps. Thus the number of discrete time steps satisfies

$$N + 1 = M(L - 1) + 1. \quad (15)$$

These relationships are depicted graphically in Fig. 1. So the optimization problem of (12) can be discretized (with the help of the trapezoidal integration rule) into into the following problem.

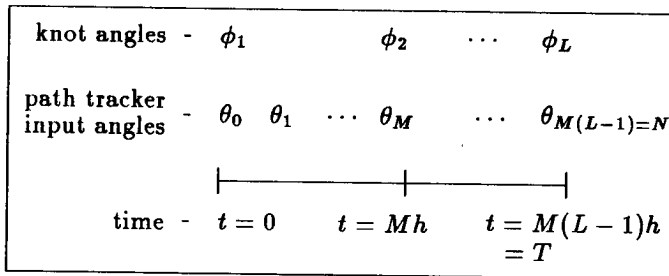
$$\min \left[\frac{1}{2}(\theta_0''')^2 + \sum_{i=1}^{N-1} (\theta_i''')^2 + \frac{1}{2}(\theta_N''')^2 \right] \quad (16)$$

$$\text{subject to } \theta_{M(j-1)} = \phi_j \quad (j = 1, \dots, L)$$

$$\theta_0^{(r)} = \theta_N^{(r)} = 0 \quad (r = 1, 2, 3)$$

where $\theta_0 = \phi_1$ and $\theta_{M(L-1)} = \phi_L$ are hard constraints, and the rest of the constraints are soft.

Fig. 1 - Relationships Between Network Variables



Finite difference expressions for the first three derivatives of $\theta(t)$ can be formed using Taylor series. These finite differences are as follows.

$$\theta'''(t) = [-\theta(t-2h) + 2\theta(t-h) - 2\theta(t+h) + \theta(t+2h)]/(2h^3) + O(h^5) \quad (17)$$

$$\theta''(t) = [\theta(t+h) - 2\theta(t) + \theta(t-h)]/h^2 + O(h^3)$$

$$\theta'(t) = [\theta(t+h) - \theta(t-h)]/(2h) + O(h^3)$$

Since the values of θ_0 and θ_N are hard constraints, they can be considered constants. Then the independent variables of the optimization problem are θ_i ($i = 1, \dots, N-1$). Note that since we are constraining θ_0''' and θ_N''' to zero,

they can be omitted from the objective function of (16). Then, using (17), the optimization problem of (16) can be converted into the equivalent problem

$$\min \sum_{i=1}^{N-1} (-\theta_{i-2} + 2\theta_{i-1} - 2\theta_{i+1} + \theta_{i+2})^2 \quad (18)$$

$$\text{subject to } \theta_{M(j-1)} = \phi_j \quad (j = 2, \dots, L-1)$$

$$\theta_1 = \phi_1$$

$$\theta_2 = \phi_1$$

$$\theta_{N-2} = \phi_L$$

$$\theta_{N-1} = \phi_L$$

where we have defined $\theta_{-1} \equiv \theta_0$ and $\theta_{N+1} \equiv \theta_N$. Now (18) can be written as

$$\min(\vec{\theta}^T A \vec{\theta} + \vec{b}^T \vec{\theta}) \quad \text{subject to } \vec{g}(\vec{\theta}) = 0 \quad (19)$$

where $\vec{\theta} = [\theta_1 \dots \theta_{N-1}]^T$, $\vec{g}(\vec{\theta})$ is the $(L+2)$ -element constraint vector defined by (18), and A and \vec{b} are respectively an $(n-1) \times (n-1)$ matrix and an $(n-1)$ -vector. Matrix A is a positive semidefinite matrix of bandwidth four [4] whose diagonal and first through fourth upper and lower diagonals are given as follows.

$$\text{diag} = (5 \quad 9 \quad 10 \quad \dots \quad 10 \quad 9 \quad 5)$$

$$\text{first diag} = (-2 \quad -4 \quad \dots \quad -4 \quad -2)$$

$$\text{second diag} = (-4 \quad -4 \quad \dots \quad -4 \quad -4)$$

$$\text{third diag} = (4 \quad 4 \quad \dots \quad 4 \quad 4) \quad (20)$$

$$\text{fourth diag} = (-1 \quad -1 \quad \dots \quad -1 \quad -1)$$

Vector \vec{b} is given by

$$\vec{b} = \begin{pmatrix} -4\phi_1 & -4\phi_1 & 6\phi_1 & -2\phi_1 \\ 0 & 0 & \dots & 0 & 0 \\ -2\phi_L & 6\phi_L & -4\phi_L & -4\phi_L \end{pmatrix}^T. \quad (21)$$

According to the results given by (4), (19) is solved by the dynamic system

$$\dot{\vec{\theta}} = -2A\vec{\theta} - \vec{b} - \frac{\partial \vec{g}}{\partial \vec{\theta}} (\vec{\lambda} + \vec{c} \circ \vec{g}) \quad (22)$$

$$\dot{\vec{\lambda}} = \vec{c} \circ \vec{g}$$

where $\vec{c} \circ \vec{g}$ is the $(L+2)$ -vector Hadamard product of \vec{c} and \vec{g} whose i^{th} element is given by $c_i g_i$. The element in the i^{th} row and j^{th} column of $\partial \vec{g} / \partial \vec{\theta}$ is given by $\partial g_j / \partial \theta_i$.

Since matrix A has bandwidth four, the neural network is not fully connected. In fact, the interconnections are quite sparse, which reduces the computational time of the network. Each θ neuron receives inputs only from itself and its four closest neighbors on each side, and from one

of the λ neurons. Each λ neuron receives inputs only from one of the θ neurons.

If matrix A was positive definite, we could set \vec{c} equal to the zero vector and still be guaranteed convergence. However, if A is only positive semidefinite, we need to use a nonzero \vec{c} . Even if A is positive definite, a nonzero \vec{c} will improve the convergence properties of the neural network.

It is important to note that the neural net considered in this paper may converge to a local minimum rather than a global minimum. A given trajectory planning problem, discretized into $(N+1)$ joint angles, is a function of $(N-1)$ variables (see (18)) and may have many local minima. The solution to which the neural net converges depends on the initial state of the network. Some sort of simulated annealing technique can be used in conjunction with the network described in this paper [5]. This idea results in the long computational time characteristic of annealing, but it also enables the network to find the best solution among many local minima.

The annealing-type method which is suggested in this paper is as follows. Once the network converges to a local minimum, the network state is perturbed in a random direction and by a random magnitude. Then the network dynamics are reactivated, and another local minimum is found. During this process, the algorithm keeps track of the best solution. After a predetermined number of local minima are found, the algorithm terminates and the solution with the lowest energy is accepted as the best solution.

This method is thus a cross between an exhaustive search and simulated annealing. It is not an exhaustive search, because a relatively few number of initial states are chosen from which the network converges to a minimum. But it is not simulated annealing either, because the network energy (in this case the integral of the square of the jerk) always decreases and there is no cooling schedule.

IV. SIMULATION RESULTS

The neural network proposed in the previous section was simulated on a Sun-4 Workstation in the C language. Three multiple-knot joint trajectories were calculated using the simulated neural network. Each joint trajectory has eight evenly spaced knots, corresponding to the examples given in previous work [8, 13, 16]. The knots were chosen along the desired paths of the first three joints of a Unimate PUMA 560 robot. Each path length is 35 seconds. The joint space knot angles are given in Table I. Each trajectory was discretized into 71 time points. So the neural network had 81 neurons, one for each time point and one for each trajectory constraint (see (18)). The parameters shown in Fig. 1 were $L = 8$, $M = 10$, $N = 70$, $h = 0.5$ seconds, and $T = 35$ seconds. A real application would require much finer discretization, perhaps on

the order of $h = 0.01$ seconds (the controller rate). The results given in this section are only intended to demonstrate the feasibility of the proposed trajectory planning method. If $h = 0.5$ was used for a real robot, the neural network solution could be interpolated by some standard method to give joint angles at the controller rate.

Joint	Knot							
	1	2	3	4	5	6	7	8
1	10	60	75	130	110	100	-10	-50
2	15	25	30	-45	-55	-70	-10	10
3	45	180	200	120	15	-10	100	50

Table I: PUMA 560 Knot Angles (Degrees)

Plots of the three neural network-based trajectories are given in [14]. The initial state of the neural nets consisted of minimum jerk trigonometric trajectories [13], $\vec{\lambda}$ was initialized to the zero vector, and \vec{c} was a vector in which each element was 1. Note from the figures that the neural network-based trajectory does not exactly pass through the knots. If the user desired the trajectory to pass closer to the knots, the relative weights of the knot constraints can be increased [14] (see (18) - (19)). Table II shows the decrease of the jerk objective function due to the evolution of the network dynamics.

Joint	Min. Jerk Trigonometric	Min. Jerk Neural Net	Percent Decrease
1	127	106	16.5
2	44	28	36.3
3	558	462	17.2
Averages	243	199	23.3

Table II: Jerk Objective Function Values

It should be noted that the trigonometric splines have zero velocity, acceleration, and jerk at the endpoints, and pass exactly through the knots. The neural network trajectories have a small nonzero velocity, acceleration, and jerk at the endpoints, and they pass near but not exactly through the knots.

V. CONCLUSION

Minimum jerk joint trajectories have the properties of similarity to human joint movements [2] and amenability to tracking [7]. This makes them attractive choices for robotics applications in spite of the fact that the dynamics are not taken into account. Analytic formulations of minimum jerk trajectories between two points are

known [7, 17]. But if there are more than two knots, analytic solutions cannot be obtained and numerical solutions must be used.

In this paper, the minimum jerk joint trajectory formulation problem is posed as a constrained quadratic optimization problem. The joint angle domain is continuous, and the time domain is discretized at the robot controller rate. The network discussed in this paper may converge to a local minimum rather than the global minimum. The solution obtained by the network depends on the initial state of the network. An annealing-type technique is used in conjunction with the network to climb out of local minimum and find the best among many solutions. The simulation results presented in this paper verify that the network can be successfully applied to robot trajectory generation.

The neural network generated trajectories pass near but not exactly through the specified knots. While this paper has dealt specifically with minimum jerk joint trajectories, there are no theoretical limitations to applying this method to other objective functions. More specifically, minimum energy or minimum torque-change trajectories could be generated with the network discussed in this paper.

REFERENCES

- [1] J. Craig, *Introduction to Robotics*. Reading, MA: Addison-Wesley, 1989.
- [2] J. Flanagan and D. Ostry, "Trajectories of human multi-joint arm movements: evidence of joint level planning," in *Experimental Robotics I, The First International Symposium*, ed. by V. Hayward and O. Khatib, New York: Springer-Verlag, 1990.
- [3] T. Flash and N. Hogan, "The coordination of arm movements: an experimentally confirmed mathematical model," *Journal of Neuroscience*, vol. 5, pp. 1688-1703, 1985.
- [4] G. Golub and C. Van Loan, "Matrix Computations (Second Edition)," Baltimore, MD: The Johns Hopkins University Press, 1989.
- [5] W. Jeffrey and R. Rosner, "Neural network processing as a tool for function optimization," *Neural Networks for Computing*, ed. by J. Denker, New York: American Institute of Physics, pp. 241-246, 1986.
- [6] M. Kawato, Y. Maeda, Y. Uno and R. Suzuki, "Trajectory formation of arm movement by cascade neural network model based on minimum torque-change criterion," *Biological Cybernetics*, vol. 62, pp. 275-288, 1990.
- [7] K. Kyriakopoulos and G. Saridis, "Minimum jerk path generation," *IEEE International Conference on Robotics and Automation*, vol. 1, pp. 364-369, 1988.
- [8] C. Lin, P. Chang, and J. Luh, "Formulation and optimization of cubic polynomial joint trajectories for industrial robots," *IEEE Transactions on Automatic Control*, vol. 28, pp. 1066-1073, Dec. 1983.
- [9] J. Platt and A. Barr, "Constrained differential optimization," in *Neural Information Processing Systems*, ed. by D. Anderson, New York, NY: American Institute of Physics, pp. 612-621, 1988.
- [10] Z. Shiller and S. Dubowsky, "Robot path planning with obstacles, actuator, gripper, and payload constraints," *The International Journal of Robotics Research*, vol. 8, pp. 3-18, Dec. 1989.
- [11] Z. Shiller and H. Lu, "Robust computation of path constrained time optimal motions," *IEEE International Conference on Robotics and Automation*, vol. 1, pp. 144-149, 1990.
- [12] K. Shin and N. McKay, "Minimum-time control of robotic manipulators with geometric path constraints," *IEEE Transactions on Automatic Control*, vol. 30, pp. 531-541, 1985.
- [13] D. Simon and C. Isik, "Optimal trigonometric joint trajectories," *Robotica*, vol. 9, pp. 379-386, Oct. 1991.
- [14] D. Simon, "The application of neural networks to optimal robot trajectory planning," *Robotics and Autonomous Systems*, in press.
- [15] H. Tan and R. Potts, "A discrete trajectory planner for robotic arms with six degrees of freedom," *IEEE Transactions on Robotics and Automation*, vol. 5, pp. 681-690, 1989.
- [16] S. Thompson and R. Patel, "Formulation of joint trajectories for industrial robots using b-splines," *IEEE Transactions on Industrial Electronics*, vol. 34, pp. 192-199, May 1987.
- [17] Y. Uno, M. Kawato and R. Suzuki, "Formation and control of optimal trajectory in human multijoint arm movement," *Biological Cybernetics*, vol. 61, pp. 89-101, 1989.
- [18] P. Wasserman and R. Oetzel, "NeuralSource: The Bibliographic Guide to Artificial Neural Networks," New York, NY: Van Nostrand Reinhold, 1990.
- [19] X. Zhao and J. Mendel, "An artificial neural minimum-variance estimator," *IEEE Conference on Neural Networks*, vol. 2, pp. 499-506, 1988.