The Synthesis of Smooth Trajectories for Pick-and-Place Operations

JORGE ANGELES, ANDREAS ALIVIZATOS AND PAUL J. ZSOMBOR-MURRAY

Abstract—**A spline-based method of programming smooth trajctories for pick-and-place operations is introduced. Unlike continuous-path operations, which impose a unique Cartesian trajectory, an infinite number of smooth trajectories can be described between any given pick and its corresponding place configuration. The method presented here allows one the construction of a unique C^2-continuous pick-and-place trajectory with attractive features. The method begins with the mapping of the pick and the place configurations in Cartesian space into joint-coordinate space, using a general-purpose inverse kinematics package that handles singularities and redundancies. Next, a trajectory, composed of a C^2-continuous, periodic cubic spline segment, is defined between the pick and the place configurations in the joint-coordinate space. It is demonstrated that C^2-continuity will prevail in Cartesian space as well. The software implementing this method includes a graphics package, to render and animate the robot motion display, as well as an interface to an off-line programming system to realize the synthesis of the actual robot motion. Finally, details of the procedure are illustrated with a numerical example applied to a commercial industrial robot.**

I. Introduction

Industrial robots are frequently used in manufacturing for pick-and-place operations (P&PO), e.g. in workpiece palletizing, machine-tool serving and assembly operations. The replacement of teach-mode programmed P&PO with computer-programmed operations is increasing. Teach mode is inaccurate and expensive. It often separates the robot from the production line for several weeks. Therefore, the expenditure of considerable effort to develop efficient methods for computer-programmed operations is quite understandable. Computer-programmed methods fall within the realm of trajectory planning, a subject of considerable interest [1], [2]. The approach introduced in the foregoing references is one of defining piecewise interpolation polynomials over wide ranges of motion to represent the joint displacement functions of the manipulator. This approach, used by Shahinpoor and Abdel-Rahman [3] to synthesize third-and fourth-order polynomials, requires the solution of up to nine simultaneous linear equations. The problem of trajectory planning, as it pertains to P&PO, has been addressed by Brooks [4], who considered the problem of obstacle avoidance, by introducing the concept of freeways in both the Cartesian- and the joint-coordinate space, while avoiding the difficulties entailed by the find-path method proposed by Lozano-Pérez [5]. Proposed in this paper is an alternate approach to trajectory planning that is suited to P&PO. It is based on the concept of spline synthesis introduced in [6] and produces smooth trajectories in both the joint- and the Cartesian-coordinate space, while considering end kinematic smoothness constraints. The method requires only two kinematic inversions, one at each end of the two prescribed end positions–the pick and the place positions. Since the programming takes place at the joint–coordinate level, and feasible mappings of obstacles into the joint–coordinate space are still under research, collisions are avoided by the introduction of a graphic display of the Cartesian trajectory, which is done using a graphics package for robot-motion animation [7].

Manuscript received December 26, 1986; revised September 28, 1987. This work supported in part by NSERC Grant #A4532 and in part by FCAR Grant #EQ 3072.

The authors are with the Department of Mechanical Engineering, McGill University, 3480 University St., Montreal, PQ, Canada H3A 2A7.

IEEE Log Number 8718790.

The reason why cubic spline functions are used here is manifold. First, the problem of producing a smooth motion between two specified positions of a body at rest is an old one, that has been studied extensively in connection with cam-follower mechanisms [8]. The usual approach to the synthesis of a smooth motion between two dwell positions of the follower in this type of mechanisms has been to resort to simple formulae for harmonic, cycloidal or polynomial motions. The drawback of the first two of those motion types is that they bear a very limited number of parameters allowing one to meet end conditions on position, velocity and acceleration. For instance, harmonic motions allow only for two parameters, namely, the amplitude and the phase, thereby making it impossible to satisfy conditions of zero velocity and acceleration at the end points. Cycloidal motions also contain two parameters, but have the property of allowing to meet end conditions of vanishing of both velocity and accelerations. Polynomial motions are defined as a single polynomial of degree n, on the independent variable—the angle of rotation of the cam disk in this case, thereby introducing up to $n+1$ coefficients, which thus allow one to meet up to $n+1$ independent conditions. This is a very attractive feature of polynomial motions, but the numerical problem of computing the aforementioned coefficients is very unstable (see, e.g., [9], for a discussion of this issue), as n increases. This drawback of polynomial–interpolation methods is the reason why current interpolation schemes are based on piecewise polynomials of a rather low degree, which takes us naturally into the realm of splines. In fact, spline functions have been successfully applied in the optimization of cam mechanisms [10]. Surprisingly, spline functions are not very popular in the domain of robot motion planning. In fact, the only references that the authors could find in this respect were [11], [12]. However, the method proposed in the first of the aforementioned references is different from the one proposed here. Indeed, Edwall, Pottinger and Ho use cubic/quartic splines whereas the method proposed here requires only cubic splines, thereby reducing the computational burden. Moreover, Edwall *et al.* require solving a system of equations every time they plan a trajectory. With the present method, no equation solving is required, for the trajectory is synthesized from scaling of a suitably defined *normal spline*. Furthermore, the said authors treated only positioning, whereas both positioning and orientation are treated here. On the other hand, Lin and Chang used a combination of X- and quartic splines to interpolate a trajectory in the Cartesian space. This requires, clearly, a kinematic inversion at each trajectory point, which, together with the nature of splines they used, amounts to a substantial increase in the underlying algorithm's computational complexity.

Finally, the synthesized trajectory is realized on an actual robot using an interface to an off-line programming system [13]. This interface is so designed that the user receives a failure message whenever the synthesized trajectory violates the position and speed limits on the joint motions. Moreover, for given trajectory and robot data, the interface computes the minimum time that is physically possible, given the joint–rate limits.

II. Problem Formulation

Let the n joint coordinates of any given robot manipulator be the components of the n-dimensional vector θ and let the Cartesian coordinates be components of vector $\boldsymbol{x}$. Note that, contrary to the common practice of defining $\boldsymbol{x}$ as a six-dimensional vector for the most general case of three-dimensional rigid-body motions, a seven-dimensional vector representation is adopted, which is numerically more stable than its six-dimensional counterpart. In fact, the rotation of a rigid body in three-dimensional space is uniquely defined by the two linear invariants of the associated rotation tensor $\boldsymbol{Q}$, namely, its vector and its trace, henceforth

0018-9472/88/0100-0173$01.00

represented by vect(Q) and tr(Q), respectively. If the unit vector parallel to the axis of rotation and the angle of the said rotation are denoted by e and ϕ, respectively, the foregoing invariants are given by

$$\text{vect}(Q) = e \sin\phi \tag{1a}$$

$$\text{tr}(Q) = 1 + 2\cos\phi. \tag{1b}$$

Alternatively, the four invariant Euler parameters could have been used, but they are quadratic functions of Q, and hence they are more expensive to compute. The remaining three scalar components of vector x are simply the Cartesian coordinates of one particular point P of the end effector. Of the seven components of vector x defined in the foregoing, the first four define the orientation, whereas the remaining ones define the position of one representative point of the end effector. The Cartesian trajectory sought is that defined by an ordered set of values of x, where the ordering parameter is the real variable t, denoting time. Moreover, t will be arbitrarily defined as zero at the initial or pick configuration, while $t = T$ at the final or place configuration. It is assumed that T is known, additionally. The values of $x(t)$ at $t = 0$ and $t = T$ will be denoted by x_I and x_F, respectively, and are assumed to be known as well. Thus

$$x(0) = x_I,\ x(T) = x_F. \tag{2a}$$

The trajectory sought is required to satisfy the smoothness end conditions given by

$$\dot{x}(0) = \dot{x}(T) = 0,\ \ddot{x}(0) = \ddot{x}(T) = 0. \tag{2b}$$

The problem thus can be stated as: Find a trajectory in the joint-coordinate space, $\theta = \theta(t)$, which will produce a smooth Cartesian trajectory $x(t)$ and satisfies (2a) and (2b).

The solution to the foregoing problem is described in the next section.

III. Proposed Solution

The solution proposed here is based on the following: where and when the joint velocity and acceleration vectors, $\dot{\theta}$ and $\ddot{\theta}$, vanish so do the Cartesian velocity and acceleration vectors, $\dot{x}$ and $\ddot{x}$.

The foregoing result is readily verified. Let $J(\theta)$ denote the Jacobian matrix relating the Cartesian and the joint velocities, i.e.,

$$\dot{x} = J\dot{\theta}. \tag{3}$$

Upon differentiation of (3) with respect to time, the following is obtained:

$$\ddot{x} = J\ddot{\theta} + \dot{J}\dot{\theta}. \tag{4}$$

From (3) and (4) it is apparent that $\dot{\theta} = 0$ implies that $\dot{x} = 0$, whereas $\ddot{x} = 0$ is implied by $\dot{\theta} = 0$ and $\ddot{\theta} = 0$. It is emphasized that the foregoing relations hold even when J is rank deficient, i.e., these relations hold even at singular configurations.

Now let θ_I and θ_F be the values attained by the joint-coordinate vector θ at the end points of the Cartesian trajectory sought, i.e.,

$$\theta_I = \theta(x_I),\ \theta_F = \theta(x_F). \tag{5}$$

Treating θ as a function of time, (5) can be rewritten as

$$\theta(0) = \theta_I,\ \theta(T) = \theta_F. \tag{6}$$

The computation of θ_I and θ_F, for given values of x_I and x_F, is known as the inverse kinematic problem. Its solution requires finding the roots of a nonlinear algebraic system, which is possible to do explicitly only for certain types of manipulators [1], [14], [15]. More recently, numerical schemes applicable to manipulators of arbitrary architecture have appeared in the literature [16]–[20]. Thus, there is no particular difficulty in computing θ at the end points of the desired trajectory. The problem of interest can now be restated as follows: Determine a smooth joint trajectory $\theta(t)$, for $t \in [0, T]$, which verifies (6) as well as the given smoothness conditions

$$\dot{\theta}(0) = \dot{\theta}(T) = 0,\ \ddot{\theta}(0) = \ddot{\theta}(T) = 0. \tag{7}$$

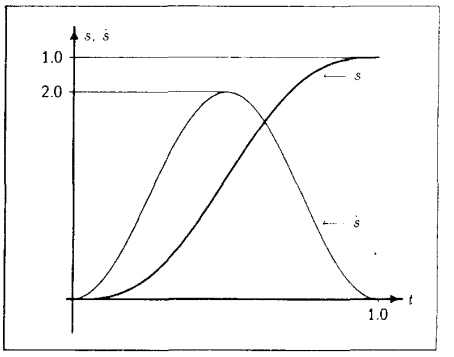
This problem is purely geometric, viz. to determine a smooth curve joining the lines $\theta_i = \theta_i(0)$ and $\theta_i = \theta_i(T)$, which is tangent to these lines, and joins them with zero curvature, at points $(0, \theta_i(0))$ and $(T, \theta_i(T))$, in the $t - \theta_i$-plane, for $i = 1, \cdots, n$. The foregoing geometric problem is solved by representing the said curve as a segment of a periodic cubic spline function. In fact, the arising cubic spline has the optimality property of minimizing the Euclidean norm, F, of its second derivative, among all smooth functions that interpolate a set of points $\{t_i, \theta_i\}_1^N$, in the t-θ plane. The said norm is defined as

$$F = \int_0^T \ddot{\theta}^2(t)\, dt.$$

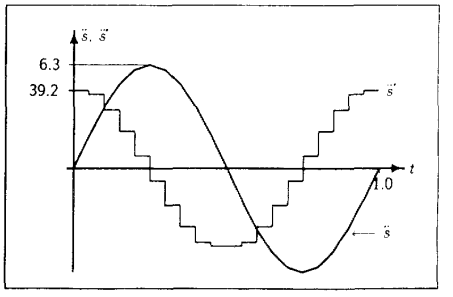
The aforementioned optimality property has the following kinematic interpretation: Among all curves passing through a set of points $P_1, P_2, \cdots, P_N$ in the t-θ plane, the cubic spline is the one containing the minimum acceleration magnitude. However, in the context of this problem, only P_1 and P_N are known, the intermediate interpolation points being, as yet, undetermined. These are found, in turn, by prescribing a harmonic distribution of $\ddot{\theta}_i(t)$ at the sample instants $t_1, t_2, \cdots, t_N$, based on a procedure described in detail in [6]. The determination of each $\theta_i(t)$ is simplified via the introduction of the normalized spline $s(t')$, which is shown in Fig. 1. This normal spline satisfies the conditions

$$s(0) = 0,\ s(1) = 1, \tag{8a}$$

$$\dot{s} = \dot{s}(1) = 0,\ \ddot{s}(0) = \ddot{s}(1) = 0, \tag{8b}$$



(a)



(b)

Fig. 1. (a) Normal spline and its first-time derivative. (b) Second- and third-time derivatives.

where the argument t' is defined as

$$t' = t/T. \tag{8c}$$

Hence, each $\theta_i(t)$ is obtained by a simple scaling of $s(t')$ and its argument, namely as

$$\theta_i(Tt') = \theta_i(0) + [\theta_i(T) - \theta_i(0)]s(t'). \tag{9}$$

The smoothness of the trajectories thus produced is guaranteed by that of the normal spline. This is apparent from the plots of the derivatives of the normal spline that appear in Fig. 1.

Once the joint coordinate histories $\theta_i(t)$, for $0 \leqslant t \leqslant T$ are obtained, the Cartesian trajectory $\boldsymbol{x}$ is readily computed as the solution to a direct kinematic problem.

IV. The Synthesis of Periodic Splines

This is a short account of the procedure used to compute the normal spline $s(t')$. For brevity, the prime superscript of t will be dropped in what follows. The spline is represented by

$$s(t) = a_i(t - t_i)^3 + b_i(t - t_i)^2 + c_i(t - t_i) + d_i,\ t_i \leqslant t \leqslant t_{i+1} \tag{10}$$

for $i = 1, \cdots, N-1$. A spline $s(t)$ is sought that meets (8a) and (8b). Conditions (8a) present no difficulty, but (8b) represents conditions that are not inherent to all cubic splines. However, by extending the domain of the spline to $t = 2$, by assigning to it a periodcity of 2, and by defining it to be an odd function of time, i.e.

$$s(t) = s(t+2),\ s(t) = -s(-t), \tag{11}$$

the foregoing conditions can all be met. Moreover, it is desired that the second derivative $\ddot{s}(t)$ be bounded within the whole interval of interest. For cubic splines, this derivative has a piecewise linear distribution: hence, in order to have it bounded in $[0,2]$, it is sufficient to assign its values at the supporting points $t_1, t_2, \cdots, t_N$. These values will be denoted by $\ddot{s}_i$, for $i = 1, \cdots, N$. A suitable distribution of this discrete function is

$$\ddot{s}_i = A \sin \pi t_i,\ t_i \in [0,2] \tag{12}$$

where A is the amplitude of $\ddot{s}_i$, a control parameter. The spline coefficients appearing in (10) are thus far undetermined. They are computed as [21]

$$a_i = \frac{1}{6\Delta t_i}(\ddot{s}_{i+1} - \ddot{s}_i) \tag{13a}$$

$$b_i = \frac{1}{2}\ddot{s}_i \tag{13b}$$

$$c_i = \frac{\Delta s_i}{\Delta t_i} - \frac{1}{6}\Delta t_i(\ddot{s}_{i+1} + 2\ddot{s}_i) \tag{13c}$$

$$d_i = s_i, \tag{13d}$$

with

$$\Delta t_i = t_{i+1} - t_i,\ \Delta s_i = s_{i+1} - s_i, \tag{13e}$$

s_i being the ordinate value at $t = t_i$, i.e., $s_i \equiv s(t_i)$, and $i = 1, \cdots, N-1$. Further definitions are $\dot{s}_i \equiv \dot{s}(t_i)$ and $\ddot{s}_i \equiv \ddot{s}(t_i)$.

Moreover, from the assumed periodicity of $s(t)$, one has

$$s_1 = s_N,\ \dot{s}_N = \dot{s}_1,\ \ddot{s}_N = \ddot{s}_1 \tag{14}$$

where the third relation is indeed verified by the assigned values appearing in (12). Now, the $(N-1)$-dimensional vectors $\boldsymbol{s}$ and $\ddot{\boldsymbol{s}}$ are defined as

$$\boldsymbol{s} = [s_1, \cdots, s_{N-1}]^T,\ \ddot{\boldsymbol{s}} = [\ddot{s}_1, \cdots, \ddot{s}_{N-1}]^T. \tag{15}$$

The continuity conditions on $s(t)$, $\dot{s}(t)$, and $\ddot{s}(t)$ thus lead to the following:

$$\boldsymbol{A}\ddot{\boldsymbol{s}} = 6\boldsymbol{C}\boldsymbol{s}, \tag{16a}$$

where $\boldsymbol{A}$ and $\boldsymbol{C}$ are $(N-1)\times(N-1)$ matrices that will be defined presently. First, Δt_i is assumed to be constant for $i = 1, \cdots, N-1$, and is denoted by τ. One then has [21]

$$\boldsymbol{A} = \tau \begin{pmatrix} 4 & 1 & 0 & 0 & \cdots & 1 \\ 1 & 4 & 1 & 0 & \cdots & 0 \\ 0 & 1 & 4 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 & 4 & 1 \\ 1 & 0 & 0 & \cdots & 1 & 4 \end{pmatrix} \tag{16b}$$

and

$$\boldsymbol{C} = \tau' \begin{pmatrix} -2 & 1 & 0 & 0 & \cdots & 1 \\ 1 & -2 & 1 & 0 & \cdots & 0 \\ 0 & 1 & -2 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 & -2 & 1 \\ 1 & 0 & 0 & \cdots & 1 & -2 \end{pmatrix} \tag{16c}$$

where $\tau' \equiv 1/\tau$.

Clearly, matrix $\boldsymbol{C}$ is singular, and hence $\boldsymbol{s}$ cannot be computed from (16a) (Obviously, a function cannot be defined solely by its second derivative). However, s_1, and hence s_N, can be arbitrarily specified to be 0. From the prescribed values of $\{\ddot{s}_i\}_1^n$ of (12), moreover, $\ddot{s}_1$ and $\ddot{s}_N$ also vanish. If the foregoing zero values are dropped from vectors $\boldsymbol{s}$ and $\ddot{\boldsymbol{s}}$, then these reduce to the following $(N-2)$-dimensional vectors:

$$\boldsymbol{s} = [s_2, \cdots, s_{N-1}]^T,\ \ddot{\boldsymbol{s}} = [\ddot{s}_2, \cdots, \ddot{s}_{N-1}]^T. \tag{17}$$

Hence, $\boldsymbol{A}$ and $\boldsymbol{C}$ are correspondingly reduced to $(N-2)\times(N-2)$ matrices having the forms

$$\boldsymbol{A} = \tau \begin{pmatrix} 4 & 1 & 0 & 0 & \cdots & 0 \\ 1 & 4 & 1 & 0 & \cdots & 0 \\ 0 & 1 & 4 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 & 4 & 1 \\ 0 & 0 & 0 & \cdots & 1 & 4 \end{pmatrix} \tag{18a}$$

and

$$\boldsymbol{C} = \tau' \begin{pmatrix} -2 & 1 & 0 & 0 & \cdots & 0 \\ 1 & -2 & 1 & 0 & \cdots & 0 \\ 0 & 1 & -2 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 & -2 & 1 \\ 0 & 0 & 0 & \cdots & 1 & -2 \end{pmatrix}. \tag{18b}$$

Now, matrix $\boldsymbol{C}$ is not only invertible, but also negative definite and tridiagonal. Moreover, its diagonal and neighboring lines have a very simple pattern that allows a very fast and inexpensive solution of system (16a). Hence, vector $\boldsymbol{s}$ can be readily computed as

$$\boldsymbol{s} = \tfrac{1}{6}\boldsymbol{C}^{-1}\boldsymbol{A}\ddot{\boldsymbol{s}}. \tag{19}$$

The normal spline shown in Fig. 1 was obtained with $N = 40$, having adjusted A of (12) so as to render unity the maximum value of s. Under these conditions, $A = 5$.

V. Example

In this example a smooth Cartesian trajectory is synthesized for a PUMA robot, whose architecture is described by the Hartenberg-Denavit parameters appearing in Table I. The pick and the place positions are given by the Cartesian coordinates

TABLE I
HARTENBERG-DENAVIT PARAMETERS FOR PUMA ARM

Joint	α(degrees)	d(mm)	a(mm)
1	−90	0	0
2	0	0	431.8
3	90	125.4	19.05
4	−90	431.8	0
5	90	0	0
6	0	0	0

TABLE II
CARTESIAN COORDINATES AT THE PICK- AND THE PLACE CONFIGURATIONS

Coordinate	Pick Position	Place Position
$e_x \sin\phi$	0.68	0.18
$e_y \sin\phi$	−0.49	0.29
$e_z \sin\phi$	−0.19	0.33
$\cos\phi$	−0.52	−0.88
p_x(mm)	−94.07	359.28
p_y(mm)	125.40	−352.23
p_z(mm)	−470.55	295.46

TABLE III
COMPUTED JOINT COORDINATES AT THE PICK- AND THE PLACE CONFIGURATIONS

Joint	Pick Position (degrees)	Place Position (degrees)
1	0	150
2	45	−100
3	200	−10
4	−100	100
5	30	−70
6	10	170

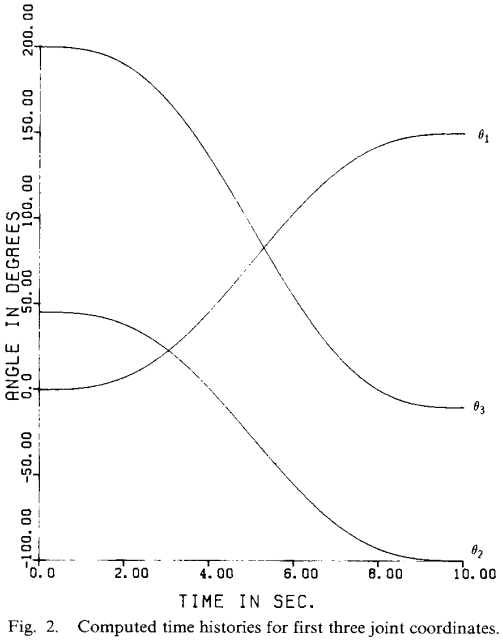


Fig. 2. Computed time histories for first three joint coordinates.

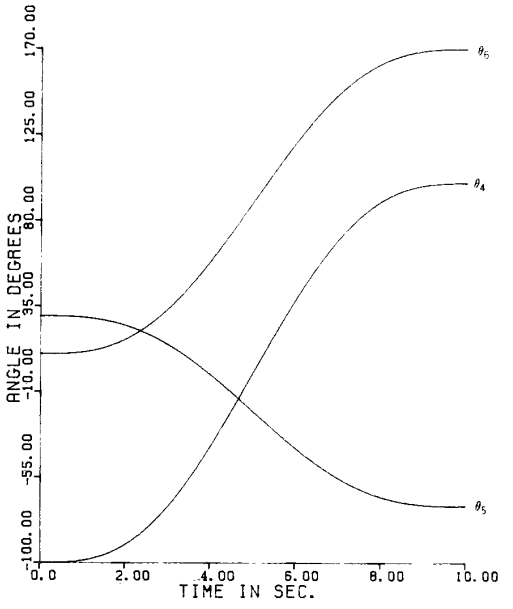


Fig. 3. Computed time histories for last three joint coordinates.

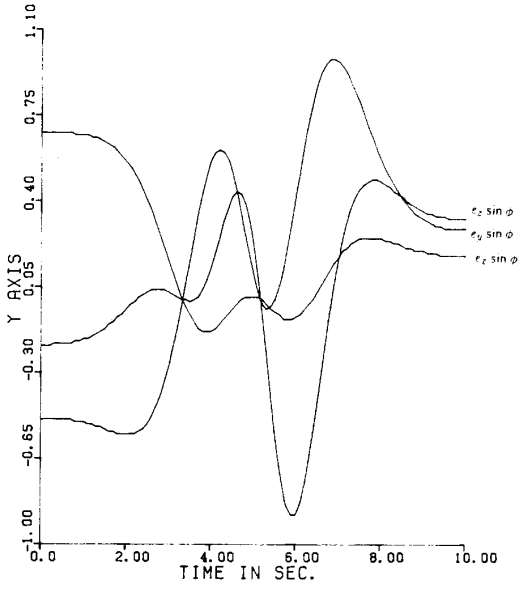


Fig. 4. Computed Cartesian components of vect(Q).

shown in Table II. The joint coordinates producing the given Cartesian coordinates at the pick and the place positions were computed using the KINVERS package, which produces kinematic inversions of robot manipulators of arbitrary architecture. It is based upon the Newton–Gauss method for nonlinear least-square solutions [19]. The results appear in Table III.

The procedure presented here was applied to synthesize the joint–coordinate trajectory appearing in Figs. 2 and 3. with $T = 10$ s. Fig. 2 shows the time histories obtained for the first three joint coordinates; Fig. 3 shows the remaining ones. A direct kinematic analysis produced the Cartesian coordinates appearing in Figs. 4–6. The three components of vector $e \sin\phi$

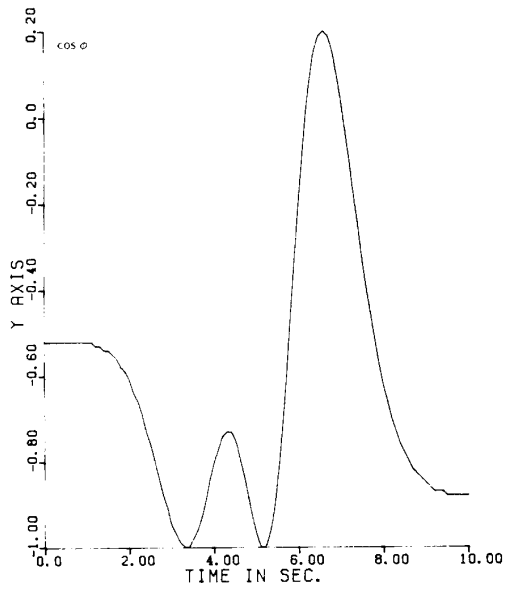


Fig. 5. Computed time history of $\cos\phi$.

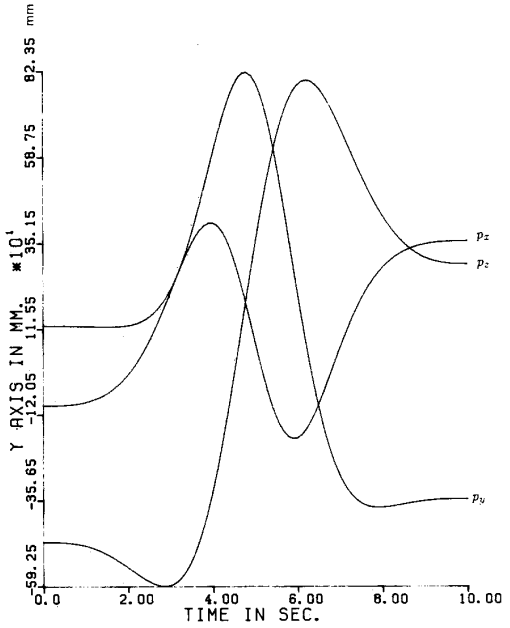


Fig. 6. Computed Cartesian components of position vector of P.

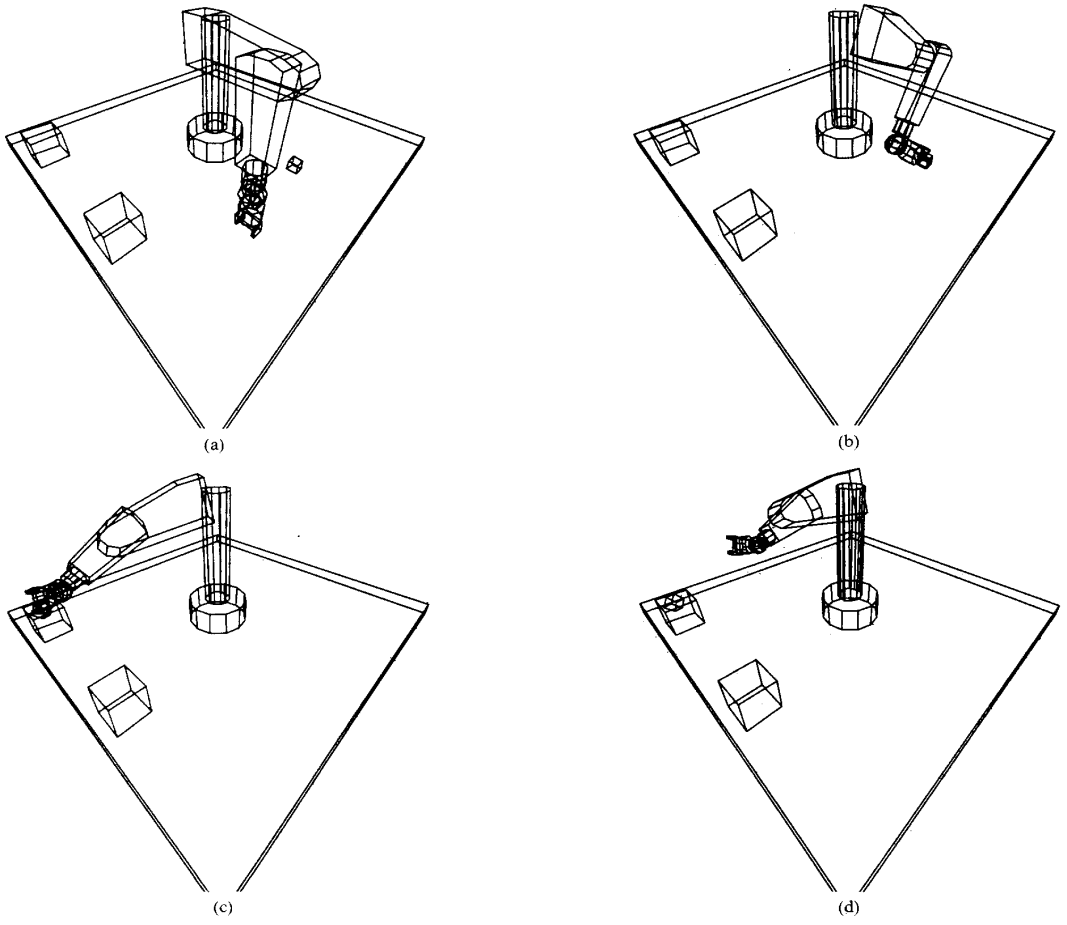


Fig. 7. Graphical display of manipulator's synthesized pick-and-place trajectory (a) Manipulator in home position. (b) Pick configuration. (c) Place configuration. (d) Back to home position.

are shown in Fig. 4, and $\cos\phi$ is shown in Fig. 5. Fig. 6 shows the three components of the position vector of point P of the end effector. The point P of the end effector that was chosen to define its position is simply the point of intersection of the three wrist axes.

From the plots of Figs. 2 and 3, one can readily verify that the computed joint-coordinate histories lie within the range of motion of the joint angles of the manipulator under consideration, namely [14]

$$-160° \leqslant \theta_1 \leqslant 160°,\ -225° \leqslant \theta_2 \leqslant 45°,\qquad -45° \leqslant \theta_3 \leqslant 225°$$

$$-170° \leqslant \theta_4 \leqslant 170°,\ -135° \leqslant \theta_5 \leqslant 135°,\quad -170° \leqslant \theta_6 \leqslant 170°.$$

Finally, the Cartesian trajectory was rendered in graphical form using a robotics-oriented graphic simulator. A few frames of this rendering appear in Fig. 7, in which (a) is the robot's *home* position, (b) is the robot's pick configuration, (c) is the robot's place configuration, and (d) is the robot on its way back to its home position.

VI. Conclusion

A method was presented that allows the synthesis of smooth trajectories for manipulators of arbitrary architecture, given the two sets of Cartesian coordinates defining the pick and the place positions. The method is based upon the concept of periodic-spline synthesis. Periodic splines proved to be readily implementable and applicable to solve the problem addressed in this paper. The procedure described here involves only two kinematic inversions, one at each end of the trajectory under study. Intermediate points of the trajectory are computed by direct kinematics, which is a rather simple task. The overall procedure, to be applicable in real time, requires, then, an efficient kinematic inversion, which does not appear to be a major problem, for most commercial robot manipulators allow an explicit inversion. More complex architectures will require an iterative inversion, like those reported in the literature, and listed in the references. Since no kinematic inversion is needed at intermediate points, singularities of the Jacobian matrix, if they ever appear, present no drawbck. Moreover, the procedure can be applied to redundant manipulators as well. The computer programming of P&PO using the method proposed here can be implemented in a matter of a few hours—less than one man-day of work, which represents considerable savings with respect to the teach-mode programming.

References

[1] R. P. Paul, *Robot Manipulators: Mathematics, Programming, and Control*, Cambridge, MA: The MIT Press, 1981.

[2] M. Brady, "Trajectory planning," in Brady M. *et al.*, Eds., *Robot Motions*, Cambridge, MA: The MIT Press, pp. 221–243.

[3] M. Shahinpoor and Z. Abdel-Rahman, "Generalized algorithm for the 4-3-4 N-axis robotic trajectory," *J. Robotic Systems*, vol. 1, no. 4, pp. 395–420, Aug. 1984.

[4] R. A. Brooks, "Planning collision-free motions for pick-and-place operations," *The International J. Robotics Research*, vol. 2, no. 4, pp. 19–44, Winter 1983.

[5] T. Lozano-Pérez, "Automatic planning of manipulator transfer movements," *IEEE Trans. Syst. Man Cybern.*, vol. SMC-11, no. 10, pp. 681–698, Oct. 1981.

[6] J. Angeles, "Synthesis of plane curves with prescribed local geometric properties using periodic splines," *Computer-Aided Design*, vol. 15, no. 3, pp. 147–155, May 1983.

[7] F. Kahloun, "A graphic simulator for a robotic workcell," M. Eng. Thesis, Department of Electrical Engineering, McGill University, Montreal, 1987.

[8] F. Y. Chen, *Mechanics and Design of Cam Mechanisms*, New York: Pergamon Press, 1982.

[9] G. Dahlquist and Å. Björck, *Numerical Methods*, Englewood-Cliffs, NJ: Prentice-Hall, 1974.

[10] J. Angeles and C. López-Cajún, "Optimal synthesis of cam mechanisms with oscillating flat-face followers," to appear in *Mechanism and Machine Theory*, 1987.

[11] C. W. Edwall, C. Y. Ho and H. J. Pottinger, "Trajectory generation and control of a robot arm using spline functions," *Robots VI*, pp. 421–445, 1982.

[12] C. S. Lin and P. R. Chang, "Joint trajectories of mechanical manipulators for Cartesian path approximation," *IEEE Trans. Syst. Man Cybern.*, vol. SMC-13, no. 6, pp. 1094–1101, Nov./Dec. 1983.

[13] M. Hoffman, "Spline-based trajectory planner for pick-and-place operations," CVaRL-McRCIM Technical report, McGill University, Montreal, 1987.

[14] R. P. Paul, B. Shimano and E. Mayer, "Kinematic control equations for simple manipulators," *IEEE Trans. Syst. Man Cybern.*, vol. SMC-11, no. 6, pp. 449–455, June 1981.

[15] A. Bazerghi, A. A. Goldenberg, and J. Apkarian, "An exact kinematic model of PUMA 560 manipulator," *IEEE Trans. Syst. Man Cybern.*, vol. SMC-14, no. 3, pp. 483–487, May/June 1984.

[16] L. W. Tsai and A. P. Morgan, "Solving the kinematics of the most general six- and five-degree-of-freedom manipulators by continuation methods," *ASME Trans. J. Mechanisms, Transmissions, and Automation in Design*, vol. 107, no. 2, pp. 189–200, June 1985.

[17] J. Lenarčič, "An efficient numerical approach for calculating the inverse kinematics for robot manipulators," *Robotica*, vol. 3, no. 1, pp. 21–26, Jan.–Mar. 1985.

[18] A. A. Goldenberg, B. Benhabib, and R. G. Fenton, "A complete generalized solution to the inverse kinematics of robotics," *IEEE J. Robotics and Automation*, vol. RA-1, no. 1, pp. 14–20, Mar. 1985.

[19] J. Angeles, "On the numerical solution of the inverse kinematic problem," *The Int. J. Robotics Research*, vol. 4, no. 2, pp. 21–37, Summer 1985.

[20] M. Takano, "A new effective solution for inverse kinematics problem (synthesis) of a robot with any type of configuration," *J. of the Faculty of Engineering*, The University of Tokyo (B), vol. 38, no. 2, pp. 107–135, Dec. 1985.

[21] H. Späth, *Spline-Algorithmen zur Konstruktion Glatter Kurven und Flächen*, 2nd. edition, Munich-Vienna: R. Oldenburg Verlag, pp. 27–44, 1978.

Fuzzy Petri Nets for Rule-Based Decisionmaking

CARL G. LOONEY

***Abstract*—The technique of fuzzy reasoning via transformations of fuzzy truth state vectors by fuzzy rule matrices is extended to Petri nets. The result is a new type of neural network where the transition bars serve as the neurons, and the nodes are conditions. Conditions may be conjuncted and disjuncted in a natural way to allow the firing of the neurons. Such conjuncting of truths is executed as generalized ANDing, i.e., MINing. Modifications are made to the usual Petri model to allow fuzzy rule-based reasoning by propositional logic. First, fuzzy values are allowed for rules and truths of conditions that appear in rules. Next, multiple copies, rather than the original, of the fuzzy truth tokens are passed along all arrows that depart a node or transition bar where the truth token resides. An algorithm is presented for reasoning via these networks, as well as a simple example for exercising the algorithm. Abduction may be done analogously by reversing all arrows and propagating truth tokens backwards.**

I. Introduction

Mathematical models of the real world have a special place in the evolution of science and engineering. However, many people have become aware that the real world is not linear quadratic and that many situations can not be modeled accurately by mathematically tractable equations [5]. Expert systems, fuzzy control systems, and other types of smart programs that have evolved more recently use data-driven decisions to adapt by branching to alternate models and submodels.

Fuzzy control systems [5] developed concurrently with expert systems and, in fact, are just small expert systems that reason forward to deduce the appropriate controls based on feedback data. These are designed to either replace, or work in conjunction

Manuscript received April 25, 1987; revised September 6, 1987.

The author is with the Department of Electrical Engineering and Computer Science, University of Nevada, Reno, NV 89557-0050.

IEEE Log Number 8717843.