論文 88-25-2-6

A Discrete-Time Trajectory Planning Method for Robotic Manipulators

(로보트 매니퓰레이터를 위한 이산시간 궤적 계획방법)

李 範 熙*

(Bum Hee Lee)

要 約

본 논문에서는 기존의 궤적 계획 방법과 달리 이산시간에서의 궤적점을 직접 구하는 방법이 연구된다. 직선 경로 추적을 고려하여 매니퓰레이터 제어를 위한 궤적점들이 한 직선상에 정확히 결정된다. 최소시간 운용을 위하여 문제구성은 두개의 인접된 서보 시간동안 공간좌표상에서의 거리의 극대화로 이루어진다. 그러한 극대화는 유연 제약조건과 토오크의 제약 조건에 의존하며, 극대화를 위해 여러가지 알고리즘들이 연구된다. 본 제안된 방법은 그 성능의 입증을 위해 VAX-11/780 컴퓨터로 모사되었다.

Abstract

In this paper, a direct method for obtaining the trajectory set points is investigated in discrete-time, which is different from the other conventional schemes. We consider the tracking of a straight line path, where the trajectory set points for manipulator control are determined exactly on the straight line path. For the purpose of the minimum-time operation of manipulators, the problem is formulated as a maximization of the Cartesian distance between two consecutive servo time instants. The maximization is subject to the smoothness and torque constraints. Several algorithms are developed and utilized to maximize the Cartesian distance. The proposed approach has been simulated on a VAX-11/780 computer to verify its performance.

I. Introduction

This paper presents a discrete time trajectory planning scheme to determine the trajectory set points on a given straight line path. In the previous efforts of trajectory planning [1-4, 6-8, 10, 11], the manipulator hand may not be on the desired path, but on the joint-interpolated polynomials. This scheme determines the trajectory set points that are exactly on the given straight line path.

The trajectory planning problem is formulated as a maximization of the Cartesian distance between two servo time instants on a given straight line path subject to the smoothness and torque constraints. Using a real time servo interval, the manipulator dynamics is taken into account in the planning process. Due to the discrete time approximations of joint velocity, acceleration, and

*正會員, 서울大學校 制御計測工學科

(Dept. of Control and Instrumentation Eng., Seoul Nat'l Univ.)

接受日字 : 1987年 8月 27日

jerk, the optimization solution involves intensive and erroneous computations which prevent it from useful applications. Thus, the optimization is realized by search algorithms. The proposed trajectory planning scheme is simulated on a VAX-11/780 computer to verify the performance.

II. Various Constraints

1. Notations

Three Euler angles, yaw ($\alpha(t)$), pitch ($\beta(t)$), and roll($\gamma(t)$)are used to represent the orientation of the hand. The position ($P(t)$), Euler angles ($\Phi(t)$), linear velocity ($v(t)$), and angular velocity ($\Omega(t)$) vectors of the manipulator hand with respect to a reference frame are defined as

$$p \triangleq (p_x, p_y, p_z)^t \; ; \Phi \triangleq (\alpha, \beta, \gamma)^T \qquad (1a)$$

$$v \triangleq (v_x, v_y, v_z)^t \; ; \Omega(t) \triangleq (\omega_x, \omega_y, \omega_z)^t \qquad (1b)$$

To initiate the discretized trajectory analysis, let us denote the sampling period for the servo control of the robot as T (usually $1 \text{ ms} \leqslant T \leqslant 28 \text{ ms}$) and $\bar{q}(k)$ to represent angular displacement $q(kT)$:

$$q(kT) \cong \bar{q}(k) \; ; k = 0, 1, \ldots \qquad (2)$$

Then we approximate the velocity, the acceleration, and the jerk, respectively, at time kT by:

$$\dot{q}(kT) \cong \frac{1}{T}(\bar{q}(k) - \bar{q}(k-1)) = \dot{\bar{q}}(k) \qquad (3)$$

$$\ddot{q}(kT) \cong \frac{1}{T^2}(\bar{q}(k) - 2\bar{q}(k-1) + \bar{q}(k-2)) = \ddot{\bar{q}}(k) \qquad (4)$$

$$w(kT) \cong \frac{1}{T^3}(\bar{q}(k) - 3\bar{q}(k-1) + 3\bar{q}(k-2) - \bar{q}(k-3)) = \bar{w}(k) \qquad (5)$$

where $w(kT)$ denotes the jerk at time $t = kT$. For simplicity, we shall drop the "bar" from the rest of the equations.

2. *Straight Line Constraint*

In the planning of straight line trajectory, the initial location [$P(k_o)$, $\Omega(k_o)$] and the fianl location [$P(k_f)$, $\Omega(k_f)$] are given for the manipulator. The straight line equation that passes through these two points is described by

$$p(k) = p(k_o) + \lambda(k) \cdot (p(k_f) - P(k_o)) \qquad (6)$$

$$\Phi(k) = \Phi(k_o) + \lambda(k) \cdot (\Phi(k_f) - \Phi(k_o)) \qquad (7)$$

where $0 \leqslant \lambda(k) \leqslant 1$, and k_o and k_f are the initial and final times, respectively. The time k_f is not fixed, but will be determined from the overall trajectory planning algorithm. The position $P(k)$ and the Euler angles $\Phi(k)$ can be augmented into a 6x1 vector and described by

$$\begin{bmatrix} p(k) \\ \Phi(k) \end{bmatrix} = N(q(k)) = (N_1(q(k)), \ldots, N_6(q(k)))^t \qquad (8)$$

where N (.) is a 6x1 nonlinear vector function depending on the configuration of a 6 degrees of freedom manipulator.

3. *Smoothness Constraint*

All discretized control set points in the joint-variable space must be within certain limits to guarantee the smoothness of the trajectory. The smoothness constraint is stipulated in three different bounds, namely, velocity bound (VB), acceleration bound (AB), and jerk bound (JB). They are given respectively as:

$$\left|\dot{q}_i(k)\right| \leqslant \epsilon_i^v \; ; \epsilon_i^v > 0, i = 1, \ldots, 6 \qquad (9)$$

$$\left|\ddot{q}_i(k)\right| \leqslant \epsilon_i^a \; ; \epsilon_i^a > 0, i = 1, \ldots, 6 \qquad (10)$$

$$\left|w_i(k)\right| \leqslant \epsilon_i^J ; \epsilon_i^J > 0, i = 1, \ldots, 6 \qquad (11)$$

where ϵ_i^v, ϵ_i^a, and ϵ_i^J are the ith element of 6-dimensional bound vectors for the manipulator. The velocity bound (VB) and acceleration bound (AB) constrain the joint actuators from exceeding the maximum limits of the velocity and acceleration. The jerk bound (JB) reduces wear of joint actuators, reduces excitation of vibrations, and constrains the smoothness. Hence, we impose VB, AB, and JB on the entire trajectory set points. Combining (9)-(11), we can obtain the joint position constraint at time t = kT:

$$q_{i,\min}(k) \leqslant q_i(k) \leqslant q_{i,\max}(k)\ ;\ i = 1,\ldots\ldots,6 \tag{12}$$

where $q_{i,\min}(k)$ and $q_{i,\max}(k)$ depend on the position, velocity, and acceleration at t = kT together with ϵ_i^v, ϵ_i^a, and ϵ_i^J.

4. Torque Constraint

In general, the dynamical behavior of a six-joint manipulator can be described by the Lagrange-Euler equations of motion as [9]

$$r(t) = D(q)\ddot{q}(t) + h(q,\dot{q}) + c(q) \tag{13}$$

where r (t) is an applied torque vector for joint actuators, c (q) is a gravitational force vector, h $(\dot{q},q)$ is the Coriolis and centrifugal force vector, and D(q) is an acceleration-related matrix. If q (k), $\dot{q}$ (k) and $\ddot{q}$ (k) are given, and treating the equations of motion as an inverse dynamics of the system, the required piecewise joint torques can be computed recursively using the Newton-Euler equations of motion as [5]

$$\tau(k) \cong [D(q(k))]\,\ddot{q}(k) + h(\dot{q}(k), q(k)) + c(q(k)) \tag{14}$$

In a simplified notation, we have

$$\tau(k) = [D_k]\,\ddot{q}(k) + h_k + c_k \tag{15}$$

where $D_k = D(q(k))$, $h_k = h(\dot{q}(k), q(k))$ and $c_k = c(q(k))$. Let us assume that the toreques generated from (15) are constrained by limits that are dependent on the joint position and velocity as,

$$\tau_{i,\min}(k) \leqslant \tau_i(k) \leqslant \tau_{i,\max}(k)\ ;\ i = 1,\ldots, 6 \tag{16}$$

Thus, (16) can be written as,

$$\tau_{i,a}^-(k) \leqslant D_{i,k}\,\ddot{q}(k) \leqslant \tau_{i,a}^+(k) \tag{17}$$

where $D_{i,k}$ represents the ith row of the matrix D_k, and $\tau_{i,a}^-$ (k) and $\tau_{i,a}^+$ (k) are written, respectively, as:

$$\tau_{i,a}^-(k) = \tau_{i,\min}(k) - h_{i,k} - c_{i,k} \tag{18a}$$

$$\tau_{i,a}^+(k) = \tau_{i,\max}(k) - h_{i,k} - c_{i,k} \tag{18b}$$

where $h_{i,k}$ and $c_{i,k}$ are the ith elements of the vectors h_k and c_k, respectively. Therefore, the joint values at each servo time instant must satisfy the straight line constraint given by (6) - (7), the smoothness constraint given by (12), and the torque constraint given by (17).

III. Speed Optimization

It is desirable to obtain the minimum time trajectory of the straight line path. Due to the discrete time formulation and approximations of joint velocity and acceleration, the optimization is formulated to maximize the Cartesian distance between two servo time instants, that is, to maximize

$$\| p(k) - p(k-1) \| \tag{19}$$

subject to the smoothness constraint, torque constraint, and the straight line constraint. Utilizing (8),

$$\begin{bmatrix} P(k) \\ \Phi(k) \end{bmatrix} \cong [\nabla N(q(k))] .\, \Delta q(k) \tag{20}$$

where $\triangle P(k) = p(k) - p(k-1)$, $\triangle\Phi(k) - \Phi(k-1)$, $\triangle q(k) = q(k) - q(k-1)$, and the elements of $[\triangle N(q(k))]$ are found to be

$$[\nabla N(q(k))]_{ij} = \frac{\partial N_i(q(k))}{\partial q_j(k)}\ ;\ i,j=1,\ldots,6 \tag{21}$$

Utilizing (6) - (7) at time t= (k-1)T and t = kT, their difference yields,

$$\begin{bmatrix} \Delta P(k) \\ \Delta\Phi(k) \end{bmatrix} = \Delta\lambda(k) \begin{bmatrix} P(k_f) - P(k_o) \\ \Phi(k_f) - \Phi(k_o) \end{bmatrix} \tag{22}$$

where $\Delta\lambda(k) = \lambda(k) - \lambda(k-1)$. Combining (20) and (22), we have:

$$[\nabla N(q(k))] \cdot \Delta q(k) \cong \Delta\lambda(k) \begin{bmatrix} p(k_f) - p(k_o) \\ \Phi(k_f) - \Phi(k_o) \end{bmatrix} \tag{23}$$

If $[\nabla N(q(k))]$ is non-singular at time $t = kT$, then

$$\Delta q(k) \cong \Delta\lambda(k)\, Q(k) \tag{24}$$

where

$$Q(k) = [\nabla N(q(k))]^{-1} \begin{bmatrix} P(k_f) - P(k_o) \\ \Phi(k_f) - \Phi(k_o) \end{bmatrix} = [Q_1(k), \ldots, Q_6(k)]^t \tag{25}$$

physically, Q (k) is a vector which relates the angular displacement of each joint with $\Delta\lambda$ (k) of a given straight line. It is notable that

$$q(k) \cong q(k-1) + \Delta\lambda(k)\, Q(k) \tag{26}$$

Combining (12) and (26), with the codnition $Q_i(k) > O$, we have:

$$\Delta\lambda_i^-(k) \leqslant \Delta\lambda(k) \leqslant \Delta\lambda_i^+(k) \;;\; i = 1, \ldots, 6 \tag{27}$$

where

$$\Delta\lambda_i^+(k) = \frac{T \cdot \dot{q}_{i,max}(k)}{Q_i(k)} \;; \tag{28}$$

$$\Delta\lambda_i^-(k) = \frac{T \cdot \dot{q}_{i,min}(k)}{Q_i(k)} \tag{29}$$

Another constraint on $\Delta\lambda$ (k) is determined from the torque constraint. Combining (17) and (26), with the codition Q_i (k) > 0, we have:

$$\Delta\lambda_{i,min}(k) \leqslant \Delta\lambda(k) \leqslant \Delta\lambda_{i,max}(k) \;; \quad i = 1, \ldots, 6 \tag{30}$$

In fact, $\Delta\lambda_{i,min}$ (k) and $\Delta\lambda_{i,max}$ (k) are the minimum and maximum constraints on $\Delta\lambda$ (k) from the ith joint torque constraint, respectively. Since $\Delta\lambda$ (k) results in the local speed at t = kT, it is desirable to formulate the problem as:

$$\text{maximize } \Delta\lambda(k) \;;\; k = 1,2,\ldots$$

subject to

$$\Delta\lambda_{i,min}(k) \leqq \Delta\lambda(k) \leqq \Delta\lambda_{i,max}(k)$$

and

$$\Delta\lambda_i^-(k) \leqq \Delta\lambda(k) < \Delta\lambda_i^+(k) \;;\; i = 1, \ldots, 6$$

The constraint on $\Delta\lambda$ (k) may not be consistent. Thus, the existence of the maximum of $\Delta\lambda$ (k) is not always guaranteed. This aspect will be discussed later. Since $\Delta\lambda_i^-$(k), $\Delta\lambda_i^+$ (k), $\Delta\lambda_{i,min}$ (k) and $\Delta\lambda_{i,max}$ (k) are obtained and derived by approximations, it is very difficult to obtain them exactly. The most significant errors are due to the derivative of the trigonometric function involving $\tan^{-1}$ [$\tan^{-1}$( )]. We now describe search algorithms for avoiding this difficulty.

IV. Search Algorithm

At the initial location of straight line path, P (k_o), Φ (k_o), v (k_o), Ω (k_o) are specified and satisfy all the physical constraints. We would like to find the control set points P (k), Φ (k) for k = 1, ... such that the robot hand will traverse the straight line. Passing the [P (ko), Φ (k_o)] values into the inverse kinematics routine, the corresponding joint angles, q (k_o), = N^{-1} (P (k_o), Φ (k_o)) are obtained. We assume, without loss of generality, that v (k_o) = Ω (k_o) = $\dot{v}$ (k_o) = $\dot{\Omega}$ (k_o) = 0 which lead to $\dot{q}$ (k_o) = $\ddot{q}$ (k_o) = 0. We further assume that w (k_o) = 0.

1. Initial Estimated Length Ratio

λ (1) can be estimated by evaluating the directional information of the given straight line. From (12), q_i (1) is bounded by $q_{i,min}$ (1) and $q_{i,max}$ (1). Depending on the straight line equation, the linear velocity components of the robot hand must be constrained. Consider

$$[v^t, \Omega^t]^t = J(q) \cdot \dot{q} \tag{31}$$

Let us denote the ij^th element of the Jacobian matrix in (31) as J_{ij}. Since the robot motion is expected to be very slow at the first servo time instant, we assume that $J_{ij}(1) \cong J_{ij}(0)$. If $v_x(1) > 0$ from the given straight line, then $\dot{q}_j(1)$ can be selected positively whenever $J_{ij}(1) > 0$, otherwise it can be selected negatively. That is, if $v_x(1) > 0$, then

$$\hat{q}_j(1) = \begin{cases} q_{j,max}(1) \text{ if } J_{lj}(0) \geqslant 0 \\ q_{j,min}(1) \text{ if } J_{lj}(0) < 0 \end{cases} \quad ; j = 1,\ldots,6 \tag{32}$$

Similarly, if $v_x(1) < 0$ then

$$\hat{q}_j(1) = \begin{cases} q_{j,max}(1) \text{ if } J_{1j}(0) < 0 \\ q_{j,min}(1) \text{ if } J_{1j}(1) \geqslant 0 \end{cases} \quad ; j = 1,\ldots,6 \tag{33}$$

From the chosen values of $\hat{q}_j(1)$, we can obtain $p_x(1)$ from the kinematic equations. Let us use a notation $\hat{\lambda}_x(1)$ to represent the estimated value of $\lambda(1)$ from the x directional requirement. Then, $\hat{\lambda}_x(1)$ is determined as:

$$\hat{\lambda}_x(1) = \frac{\hat{P}_x(1) - P_x(k_o)}{P_x(k_f) - Px(ko)} \tag{34}$$

where if $P_x(k_f) = P_x(k_o)$, then there is no constraint on $\hat{\lambda}_x(1)$. Similarly, additional constraints can be obtained along the y,z axes. Also, the given Euler angle information gives three more constraints on the $\hat{\lambda}(1)$. Then we can choose $\hat{\lambda}(1)$ as:

$$\hat{\lambda}(1) = \min \{ \hat{\lambda}_x(1), \hat{\lambda}_y(1), \hat{\lambda}_z(1), \hat{\lambda}_\alpha(1), \hat{\lambda}_\beta(1), \hat{\lambda}_\gamma(1) \} \tag{35}$$

Recall that $\Delta\lambda(1) = \lambda(1)$ by the definition of λ (k). The choice of the value of λ (1) as in (35) means that the robot motion at t = T is constrained critically by the corresponding position or Euler angle information. For the smoothness constraint, the best initial estimate of q_i (k) for k = 2,3,..., which corresponds to $\Delta\hat{\lambda}^1$ (k), is chosen to be the mid-point of the allowable position bounds as,

$$\hat{q}_i(k) = \frac{1}{2}(q_{i,min}(k) + q_{i,max}(k)) \; ; i = 1,\ldots,6 \tag{36}$$

Using $\hat{q}_i$ (k) and q_i (k-1), the initial estimate of joint velocity is found to be

$$\dot{q}(k) \cong \frac{1}{T}(\hat{q}_i(k) - q_i(k-1)) \; ; i = 1,\ldots,6 \tag{37}$$

The initial estimate of linear velocity, $\hat{v}$ (k), can be obtained by using (31) as,

$$\hat{v}(k) = J_u(\hat{q}(k))\,\dot{q}(k) \tag{38}$$

where J_u (q (k)) is the upper 3x6 submatrix of the Jacobian matrix in (31). To obtain the estimated value of $\Delta\hat{\lambda}^1$ (k) for k = 2,3,..., let $\hat{v}_p$ (k) be the projection of v (k) onto the given straight line. Then,

$$\hat{v}_p(k) \cdot T \cong \Delta\hat{\lambda}(k); \; \ell_{total} \tag{39}$$

where ℓ_{total} is the total length of the straight line path. Thus,

$$\Delta\hat{\lambda}^1(k) = T \cdot \frac{(P(k_f) - P(k_o)) \cdot \hat{v}(k)}{\| P(k_f) - p(k_o) \| \cdot \ell_{total}} \tag{40}$$

The estimate $\Delta\hat{\lambda}^1$ (k) will be used in the following search algorithm to obtain the maximum value of $\Delta\lambda$ (k).

2. *Forward Search Algorithm*

After finding the initial estimate $\Delta\hat{\lambda}^1$ (k), the objective is to find the largest value of $\Delta\lambda$ (k) on the straight line subject to the smoothness and torque constraints. This is solved by an iterative forward binary search algorithm which is used to determine all the control set points for the acceleration portion of the given straight line. The algorithm FW is outlined below.

Algorithm FW

FW1 [Initialize and loop] For each k = 1,2, . . ., perform steps FW2 to FW8.

FW2 [Compute the desired set of constraint limits] (1) Smoothness constraint as in (12) : $q_{i,min}$ (k) and $q_{i,max}$ (k), (2) Torque constraint as in (17): $\tau^-_{i,a}$ (k) and $\tau^+_{i,a}$ (k)

FW3 [Initial estimate] Find $\Delta\hat{\lambda}^1$ (k) ((35) and (40)) and set m=0. Also, set $\Delta\lambda^{low}$ (k) = $\rho_1 \cdot \Delta\hat{\lambda}^1$ (k) and $\Delta\lambda^{high}$ (k) = $\rho_2 \cdot \Delta\hat{\lambda}^1$ (k), where ρ_1 and ρ_2 are user designated variables (when k=m=1, $\Delta\lambda^{low}(1) = 0$).

FW4 [Check the global stopping criterion for $\hat{\lambda}^m$ (k)] Update the value of m and find $\hat{\lambda}^m$ (k) = λ (k-1) + $\Delta\hat{\lambda}^m$ (k). If λ^m (k) $< 1-\delta_1$, (where $\delta_1 > 0$ is a user designated variable) then go to step FW5, otherwise stop. (Trajectory planning has been completed.)

FW5 [Find Cartesian solution.] Compute $\hat{P}$ (k) and $\hat{\Phi}$ (k).

FW6 [Find the inverse kinematics solution] Find

$$\hat{q}(k) = N^{-1}(\hat{P}(k), \hat{\Phi}(k))$$

FW7 [Check estimate's constraints and adjust $\Delta\hat{\lambda}^m$ (k)] If $\hat{q}$ (k) and $\ddot{q}$ (k) satisfy the bounds in FW2,

then set $\Delta\lambda^{low}$ (k) = $\Delta\hat{\lambda}^m$ (k) and

$$\Delta\hat{\lambda}^{m+1}(k) = \frac{\Delta\lambda^{low}(k) + \Delta\lambda^{high}(k),}{2}$$

and go to step FW4.

FW8 [Check stopping criterion for $\Delta\hat{\lambda}^m$ (k)]

If $|\Delta\hat{\lambda}^{m+1}(k) - \Delta\hat{\lambda}^m(k)| < \delta_2 \cdot \Delta\hat{\lambda}^1(k)$ (where $\delta_2 > 0$ is a user designated variable) then $\Delta\lambda$ (k) = $\Delta\hat{\lambda}^m$ (k) ; k = k+1 ; and go to step FW2, otherwise go to step FW4.

3. *Determination of Break Point*

The forward search algorithm FW generates the trajectory set points which constitute the acceleration portion of the straight line. Thus, a point must be determined on the straight line so that the deceleration portion starts from the point to meet the boundary conditions at the final location. A break point is defined as a point on the given straight line at which the acceleration portion of the straight line ends and the deceleration portion of the straight line starts . This break point can be determined from the following procedure BR.

Procedure BR

BR1 [Determine a possible break point] Determine a set point P (s) such that

$$\| P(s) - P_b(r) \| + k_d \cdot \left\| \frac{P(s+1) - P(s)}{T} - \frac{P_b(r) - P_b(r-1)}{T} \right\|$$

is minimum over s and r, where the subscript b denotes the search points obtainable by applying the algorithm FW from the final point to the initial point. The number k_d is a weighting factor that measures the slope error between the forward and backward motions. Also, set the loop counter c = 0.

BR2 [Apply the MFW algorithm from the set point found in BR1.] Update the loop counter c. Treating the point P (s) as an initial position, apply the modified forward search algorithm for the deceleration motion and check whether the algorithm MFW is stopped by the global stopping criterion δ_1.

BR3 [Adjust the location of the break point.] If the search algorithm MFW is stopped by the global stopping variable δ_1, then decrease s by 1 and go to step BR2, otherwise go to step BR4.

BR4 [Determine the best break point.] If the loop counter c $\geqslant 2$, then P (s+1) is the best break point, and compute the final joint position, velocity, and acceleration. otherwise reset the loop counter c to 0, increase s by 1, and go to step BR2.

The modified forward binary search algorithm is identical to the algorithm FW except the step FW7. The step FW7 is modified in the algorithm MFW as follows:

Algorithm MFW

MFW7 [Check estimate's constraints and adjust $\Delta\hat{\lambda}^{m}(k)$]. If the joint inverse kinematics solution satisfies the bounds computed in step MFW2,

then set $\Delta\lambda^{high}(k) = \Delta\hat{\lambda}^{m}(k)$ and

$$\Delta\hat{\lambda}^{m+1}(k) = \frac{\Delta\lambda^{low}(k) + \Delta\lambda^{high}(k),}{2}$$

and go to step MFW8,

otherwise set $\Delta\lambda^{low}(k) = \Delta\hat{\lambda}^{m}(k)$ and

$$\Delta\hat{\lambda}^{m+1}(k) = \frac{\Delta\lambda^{low}(k) + \Delta\lambda^{high}(k),}{2}$$

and go to se step MFW4.

In allgorithm MFW, the δ_1 and δ_2 must be chosen such that the former generates the inverse kinematics soultion which does not satisfy the constraint limits in step MFW2, while the latter generates the inverse kinematics solution which satisfies the constraint limits in step MFW2.

4. Existence of Straight Line Trajectory

For a given straight line path, there is no general rule to guarantee the existence of the straight line trajectory with an acceleration portion followed by a deceleration portion. The straight line trajectory between two specified end points may not exist if the average speed of the robot is beyond a certain limit [3]. The failure of the algorithm FW or BR can happen for the following cases : (1) when the initial estimate $\Delta\hat{\lambda}^{1}(k)$ generates the inverse kinematics solution which does not satisfy the smoothness and torque constraints; (2) when the algorithm FW or BR is not terminated by the global stopping variable δ_1. Since either case will generate a large number in the loop counter m, the failure of the algorithm can be detected if the loop counter exceeds a prespecified number of the bisection process. Then a different planning must be used for the given straight line path. Here a procedure (Procedure ESLT) is proposed to check the existence of straight line trajectory and to obtain a different trajectory planning for a given straight line.

Procedure ESLT

ESLT1 [Check whether the algorithm FW is completed or not.] Identify the failure of the algorithm FW. When the loop counter m exceeds a specified number, apply this procedure to a given straight line path. If search points from the algorithm FW are completed and stopped by δ_1, then go to step ESLT2. Otherwise go to step ESLT3.

ESLT2 [Find a break point and apply the algorithm MFW.] Apply the procedure BR and the algorithm MFW, and stop. Straight line trajectory planning is completed.

ESLT3 [Apply the algorithm MFW from the final search point.] Apply the algorithm MFW from the final search point of the algorithm FW to the desired final point. If the algorithm MFW is completed and stopped by δ_1, then apply the procedure BR (from step BR2 to BR4). Straight line trajectory planning is completed. Otherwise go to step ESLT4.

ESLT4 [From the final search point of the algorithm MFW, start to apply the FW algorithm.] Take the final search point from the algorithm

MFW as the initial point and go to step ESLT1. Take the initial estimated $\Delta\hat{\lambda}$ (k) of the final search point as the initial estimated $\Delta\hat{\lambda}$ (k) of the forthcoming algorithm FW in step ESLT1.

The procedure ESLT generates trajectory set points with a combination of several acceleration and deceleration portions for a given straight line path. The resultant trajectory may require a longer traveling time than that of the trajectory with one break point for a given straight line path.

V. Computer Simulation

A C-language program is written on a VAX-11/780 computer to implement and evaluate the performance of the proposed trajectory planning schem for a PUMA 560 series robot arm. The trajectory planner reads in the given initial and final locations of a straight line path with the stopping constants δ_1 and δ_2 and outputs the control set points of joint position, velocity and acceleration along the given straight line path. The servo control period in this simulation is assumed to be 10ms. Numerical values in Table 1 are used in the simulation. The torque constraints which depend on the instantaneous joint position and velocity are not available from the manufacturer's specification sheet for the PUMA robot. Thus, a set of constant torque constraints has been implemented in the simulation. The torque constraints used for joint 1, joint 2, ..., joint 6 are 232nm, 375Nm, 188Nm, 94Nm, and 94Nm. Also, we implemented the smoothness constraint in Table 2. The choice of δ_1 and δ_2 is closely related to the existence of straight line trajectory for a given path. The δ_1 and δ_2 are selected to be 1 and 2, respectively, in the algorithm FW, and 0 and 1, respectively, in the algorithm MFW. If the trajectory planner started the deceleration portion from the 84th point of the trajectory set points, it produced the smallest position and Euler angle errors at the final search point among the all possible break points. Hence, the 84th point from the procedure BR is the best break point for this trajectory planning. The resultant Cartesian position error due to

Table 1. Data used in computer simulation.

Initial Cartesian Point (m)	(-0. 1400 00, 0. 560000, 0. 390000)
Initial Orientation (deg)	(0. 00000, 90. 00000, 90. 00000)
Initial Joint Vel. & Acc. (deg/sec and deg/sec^2)	$\dot{q}_i = \ddot{q}_i = 0,\ i = 1, 2, \cdots, n$
Final Cartesian Point (m)	(0. 000000, 0. 440000, 0. 480000)
Final Orientation (deg.)	(30. 00000, 60. 00000, 60. 00000)
Final Joint Vel. & Acc. (deg/sec and deg/sec^2)	$\dot{q}_i = \ddot{q}_i = 0,\ i = 1, 2, \cdots, n$
Sampling Period	10msec
Stopping Variables	$\delta_1 = 0.000001,\ \delta_2 = 0.0001$
Number of Forward Search Points	121
Number of Backward Search Points	121
Possible Break Point	at the 85th sampling point
Actual Break Point	at the 84th sampling point

Table 2. Physical constraints for computer simulation.

	Joint 1	Joint 2	Joint 3	Joint 4	Joint 5	Joint 6
ε_i^J (deg/sec^3)	700	500	2100	4000	2100	8100
ε_i^a (deg/sec^2)	45	40	75	70	90	80
ε_i^v (deg/sec)	100	95	100	150	130	110

discrete time approximation is found to be less than 1mm while the maximum Euler angle error is found to be less than 0.1 degree. Figures 1-3 show the trajectory curves for joint 1,3, and 5.

IV. Conclusion

This paper presents a discrete time trajectory planning scheme to determine the trajectory set points on a given straight line path which satisfy both the smoothness and torque constraints. A real time servo interval was used for planning the manipulator trajectory. An interative forward and backward search algorithm was developed to determine the control set points from the initial position to a break point on the straight line. Then a modified forward search algorithm was

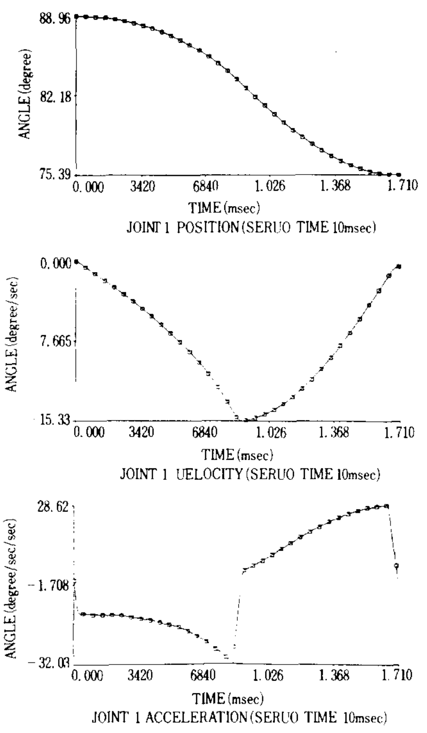

Fig. 1. Joint 1 trajectory of the straight line.

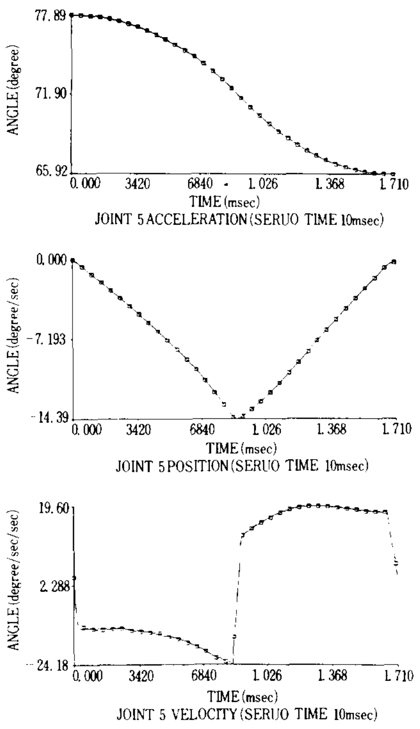

Fig. 3. Joint 5 trajectory of the straight line.

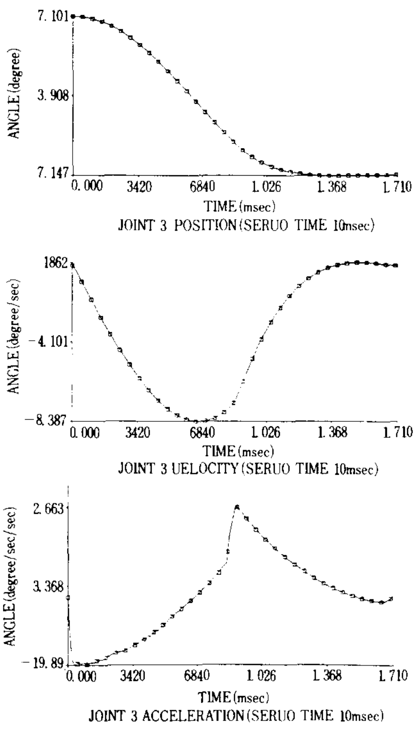

Fig. 2. Joint 3 trajectory of the straight line.

developed to determine the control set points on the straight line path from the break point to the final position. Computationally, the proposed trajectory planning provides a much simpler scheme for determining every control set point for controlling the robot arm than other conventional methods. Finally, a computer simulation was conducted to show the performance of the proposed method.

References

[1] J.E. Bobrow, S. Dubowsky and J.S. Bibson, "On the optimal control of robotic manipulators with actuator constraints," *Proc. of 1983 American Control Conf.*, San Francisco, CA, pp. 782-787. June 1983

[2] M. Brady, et al. (editors), Motion Planning, MIT Press, pp.221-243. 1982

[3] J.M. Hollerbach, "Dynamic scaling of manipulator trajectories," *Trans. ASME, J. Dynamic Syst., Meas., Contr.*, vol.106, no.1, pp.102-106, March 1984.

[4] C.S. Lin, P.P. Chang, and J.Y.S. Luh, "Formulation and optimization of cubic polynomial joint trajectories for industrial robots," *IEEE Trans. Auto. Contr.*, vol.AC-28, no.12, December 1983, pp.1066-1073.

[5] J.Y.S. Luh, M.W. Walker and R.P.C. Paul, "On-line computational scheme for mechanical manipulators," *Trans. of ASME,* J. of Dynamic Systems, Measurement, and Control, vol.120, June pp.69-76. 1980.

[6] J.Y.S. Luh and C.S. Lin, "Optimum path planning for mechanical manipulators," *Trans, ASME, J. Dynamic Syst., Meas., Contr.*, vol.102, pp.142-151 June 1981.

[7] J.Y.S. Luh and C.S. Lin, "Approximate joint trajectories for control of industrial roborts along cartesian path," *IEEE Trans. Syst., Man, Cybern.*, vol.SMC-14, no.3, pp. 444-450. May/June 1984.

[8] R.P. Paul, "Manipulator cartesian path control," *IEEE Trans. Syst., Man, Cybern.*, vol.SMC-9, no.11, pp.702-711. November 1979.

[9] R.P. Paul, Robot Manipulators : Mathematics, Programming, and Control, MIT Press, 1981.

[10] K.G. Shin and N.D. Mckay, "Minimum-time control of robotic manipulators with geometric path constraints," *IEEE Trans. Auto. Contr.*, vol.AC-30, no.6, pp.531-541. June 1985.

[11] R.H. Taylor, "Planning and execution of straight line manipulator trajectories," *IBM J. Res. Develop.*, vol.23, no.4, pp.424-436. July 1979.