[12] P. H. Winston, *Artificial Intelligence*. Reading, MA: Addison-Wesley, 1977.
[13] M. Brady, J. M. Hollerbach, T. L. Johnson, T. Lozano-Perez, and M. T. Mason, Eds., *Robot Motion: Planning and Control*. Cambridge, MA: MIT 1983.
[14] M. A. Wesley, T. Lozano-Perez, L. I. Lieberman, M. A. Lavin, and D. D. Grossman, "A geometric modeling system for automated mechanical assembly," *IBM J. Res. Development*, vol. 24, no. 1, Jan. 1980.
[15] R. A. Volz, A. C. Woo, J. D. Wolter, T. N. Mudge, J. L. Turney, and D. A. Gal, *CAD, Robot Programming and Ada*. NATO Advanced Studies Inst. on Robotics & Artificial Intelligence, Barga, Italy, June 1983.

Minimum-Time Path Planning for Robot Arms and Their Dynamics

BYUNG KOOK KIM, MEMBER, IEEE, AND KANG G. SHIN, SENIOR MEMBER, IEEE

Abstract—**Path planning of a robot arm is concerned with the derivation of a history of positions and velocities to be followed by the robot arm. Despite its close relationship to the robot arm dynamics, conventional path planning does not take into account the robot arm dynamics, leading to the possible underutilization of the robot's capabilities. We have developed a minimum-time path-planning method in joint space with the consideration of the robot arm dynamics as well as other realistic constraints. The main differences between this method and others are 1) an absolute tolerance in the path deviation at each corner point can be specified, 2) *local* upper bounds on joint accelerations are derived from the arm dynamics so as to nearly fully utilize robot's capabilities, and 3) a set of local optimization problems—one at every local corner point—are employed to replace the global minimum-time problem, thus making the minimum-time path-planning problem simpler and easier to solve. As a demonstrative example we have applied the method to the path planning of the first three joints of the Unimation PUMA 600 series robot arm using its simulator on a DEC VAX-11 / 780. The example has indeed shown significant improvements in the total traveling time in addition to the ease and simplicity obtained from the decomposition of the global problem into a set of local optimization problems.**

I. Introduction

HIGH POTENTIAL of robotics in industrial automation for both productivity increase and improvement of product quality can be fulfilled by maximally utilizing the capabilities offered by the growing number of flexible industrial robots. The optimal control of the robots is thus a key to the success in industrial automation. However, the optimal control of robots, more specifically robotic arms, is known to be a very difficult problem due to a) their nonlinear, coupled dynamics; b) physical constraints, such as limits on the angular velocity and on the torque/force generated by each joint actuator [1]; and c) the danger of colliding with other objects in the workspace. The development of the optimal control of robot arms[1] has been significantly hampered by these difficulties, and, therefore, there are only a few known attempts in this direction, leading to suboptimal control solutions with performance indices of linear quadratic functional [2], minimum-time [3], and weighted time–fuel [4].

Recognizing the difficulties in the direct optimal control of the robot arm, we have usually relied on an alternative approach to the robot arm control problem using a two-stage optimization, namely off-line path planning followed by on-line path tracking. Depending upon the system objective, both the path-planning and the path-tracking problems have to be solved by optimizing suitable criteria subject to the associated arm dynamics and other constraints. Conventionally, path tracking takes the arm dynamics into consideration [5], [6], whereas path planning does not, even though it calculates the timing of joint positions and velocities which are closely related to the arm dynamics. This fact may be justified in view of the difficulty in obtaining path-planning solutions, but the resulting timing of positions and velocities may force the robot to be underutilized, e.g., the robot may be driven slower than necessary (see [14] for a detailed discussion on this).

Usually the desired path is specified by a set of straight line segments connecting the corner points in Cartesian space, where each corner point represents position and orientation of the end-effector, and the constraints are given on the torque/force and angular velocity of each

Manuscript received February 10, 1984; revised July 9, 1984 and October 20, 1984. This work was supported in part by the NSF grant No. ECS-8409938 and the US AFOSR Contract No. F49620-82-C-0089. Any opinions, findings, and conclusions or recommendations in this paper are those of the authors and do not necessarily reflect the view of the funding agencies.

B. K. Kim was a visiting scholar in the Department of Electrical Engineering and Computer Science, University of Michigan, Ann Arbor. He is now with Woojin Instrument Ind. Co., Ltd. Seoul, Korea.

K. G. Shin is with the Department of Electrical Engineering and Computer Science, The University of Michigan, Ann Arbor, MI 48109, USA.

[1] To distinguish this from most conventional controls of robot arms, we call this the *direct optimal control* of robot arms in this paper.

0018-9472/85/0300-0213$01.00

joint. Note that these corner points are determined by a task planner, considering both the application task at hand and the avoidance of collision with other objects in the workspace. Thus one must either convert the Cartesian path to joint paths and then solve the path-planning problem in joint space, or convert the joint force/torque bounds to the Cartesian ones and then solve the Cartesian path-planning problem. The latter conversion is usually done experimentally, since analytical conversion is too complicated [12]. The former conversion is performed point-to-point for sufficiently many points on the Cartesian path, and then the joint paths can be approximated with, for example, spline functions connecting the discrete points.

An on-line Cartesian path-tracking scheme was proposed by Paul [7] but requires too much computation to implement in real-time. The minimum-time path-planning problem was investigated by Luh and Lin [8], where they derived a method for obtaining a time history of positions[2] and velocities along the path with a minimum traveling time under the constraints of Cartesian limits on linear and angular velocities and accelerations. In the Cartesian space path planning, the motion on the path segment is well defined in the context of application, but—despite the existence of efficient inverse kinematics algorithms, e.g. [15]—its computational demand is too high for real-time implementation[3] due to the required transformation of Cartesian coordinate points to joint coordinate points, and it also becomes difficult to handle the case when the robot arm is at degenerate positions.

Due to its ability to be directly implemented, the path planning in joint space is also attractive. Paul [9] suggested a simple approach that eliminates stopping at each transition from one segment to another. The time for transition is fixed and is determined to allow velocity changes from maximum to minimum and vice versa. Lin *et al.* [10] used an approximation with cubic spline functions and developed a time-scheduling algorithm by minimizing the total traveling time subject to constant limits on speed, acceleration, and jerk for each point. Taylor [11] proposed an iterative algorithm to compute sufficient number of intermediate knot points in joint space so that the transformation error—the difference between the original Cartesian path and the path obtained from transforming back the joint path that is formed by a point-to-point transformation (from Cartesian to joint spaces) and linear interpolations—may become smaller than the prescribed value. The path planning in joint space is simple and efficient, since it is limited only by maximum joint torques and velocities, and since degeneracies of the robot arm do not present any difficulty. The minimum-time path planning in joint space, which is dealt with in this paper, is particularly useful for specifying *gross motion* of the robot arm when 1) it operates in a collision-free space and 2) the minimum-time traveling is important. Also, the path planning in joint space is easily achievable since one can get close approximations in joint space to the straight line paths in Cartesian space by a) including sufficient number of intermediate knot points in the Cartesian paths (in addition to corner points), and b) employing a linear interpolation in joint space so as not to exceed the specified maximum allowable tolerance in the transformation error.

Conventional path planning in joint space uses a constant bound on the acceleration. This bound must represent the global least upper bound (GLUB) of all operating accelerations in joint space so as to enable the robot arm to follow the prespecified path under any operating condition (e.g., position, orientation, and payload). It implies that the full capability of the robot arm cannot be utilized if the conventional approach is taken [14]. If the GLUB were not used, then one would have to determine whether a planned path is dynamically realizable, and if not, how to modify the path [16]. This implies that an additional validation and/or modification of a planned path is required for the conventional path planning. Moreover, the path deviations allowed at corner points are not considered explicitly, which is for the sake of simplicity of the solution rather than for the application reality.

In this paper we consider a remedy for the above drawbacks. We propose a path-planning method in joint space by minimizing the total traveling time, given a set of corner points[4] in joint space and realistic constraints, that is, those on the generated torques/forces and angular velocities and on the deviation at each corner point in the joint paths. Note that 1) limits on the generated torques/forces in place of the accelerations introduce the robot arm dynamics into path planning, thus eliminating the need of an additional validation/modification of the planned path, and 2) absolute deviation bounds at corner points express the manufacturing reality more accurately and clearly than the case with implicit bounds (e.g., a fraction of the corresponding segment). Furthermore, the path-planning method to be developed here can be decomposed into a set of local optimizations at corner points under a certain condition. A set of local bounds on acceleration, velocity, and path deviation are used in the decomposed local optimizations. Since the optimization in path planning requires bounds on accelerations, the bounds on torques are converted to a set of bounds on accelerations, each bound being valid only in the vicinity of a corner point. The conversion calls for the robot arm dynamics and therefore results in acceleration bounds that in turn yield paths that can nearly fully utilize the robot's capabilities.

This paper is organized as follows. In Section II, the minimum-time path-planning problem is defined. Torque-to-acceleration conversion of constraints is discussed in

[2] Henceforth, the term "position" is used to mean both position and orientation.

[3] Notice that there are many computation-intensive tasks to be performed in real-time for intelligent robots, e.g., visual and tactile sensing and processing.

[4] Note that there could be an extreme case where a path segment is too short to include the cruise stage. However, we do not consider here this extreme case since the intended use of the present method is concerned with relatively long path segments as in [8].

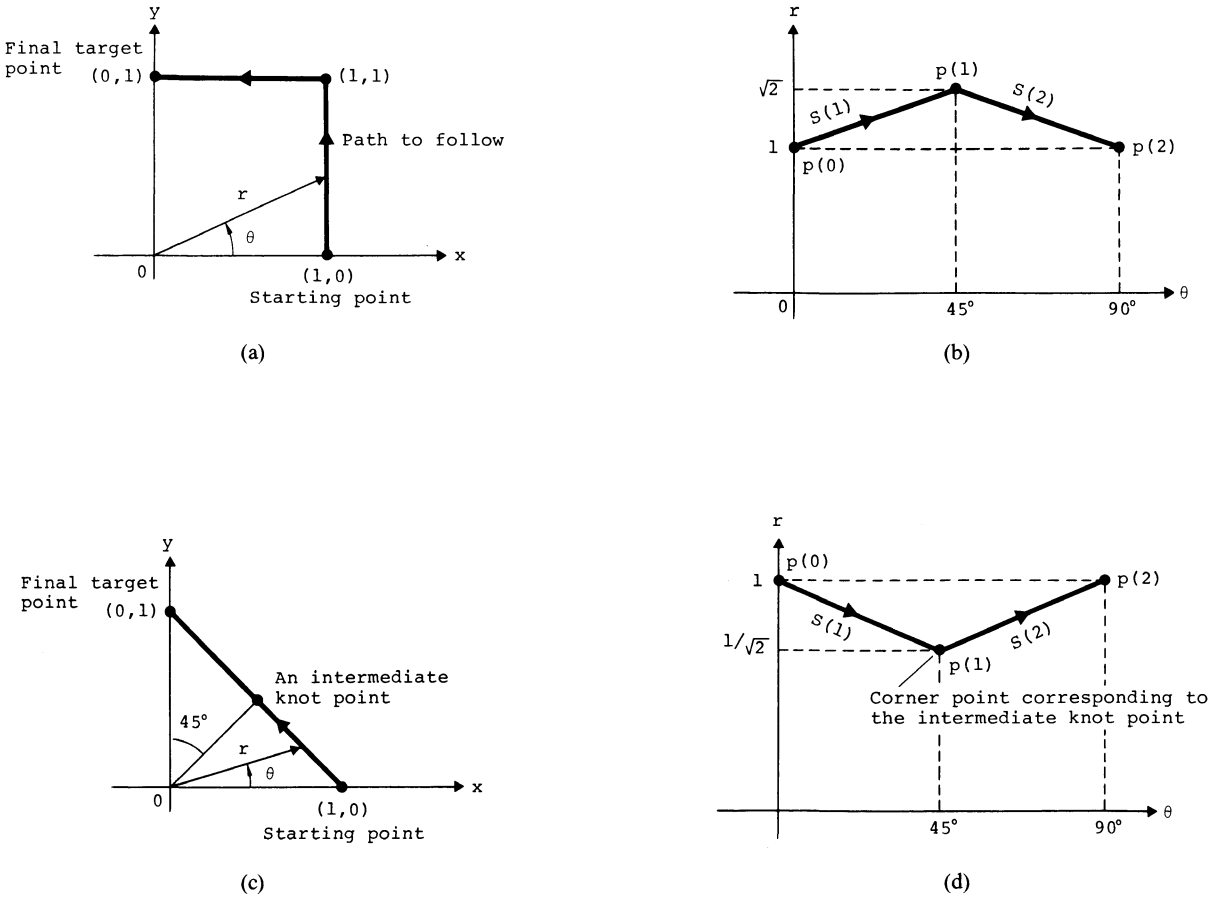

Fig. 1. Example for illustrating Cartesian paths and linearly interpolated joint paths. (a) Cartesian path generated by a task planner. (b) Joint path corresponding to (a) with a linear interpolation. (c) Cartesian path with introduction of an intermediate knot point. (d) Joint path corresponding to (c) with a linear interpolation.

Section III, and a solution to the path-planning problem is derived in Section IV. The proposed path planning is simulated on a DEC VAX-11/780 for the first three joints of the PUMA 600 series robot arm in Section V, showing the improvements in the total traveling time.

II. Problem Statement

Considering the task to be performed and interactions with the working environment [13], the task planner generates a desired (geometric) path for the robot arm in Cartesian space. The geometric path does not contain any timing information but includes only spatial positions and orientations. The set of the desired corner points $\boldsymbol{p}(i)$, $i = 0, 1, \cdots, M$, in joint space can be obtained by transforming the Cartesian corner points (i.e., output of the geometric path planner) and intermediate knot points into joint space. The intermediate knot points in the Cartesian path are added to ensure that the transformation error must be smaller than the prescribed accuracy. The prescribed accuracy is supplied by the task planner.

A path segment in joint space is formed by connecting, with a straight line, two adjacent corner points (see [18] for the same concept of a straight line path segment in joint space). In order to have a better view of a straight line path segment in joint space, consider a simple example in Fig. 1. The example robot consists of two degrees of freedom: the one is rotational and the other extensory. Suppose the desired Cartesian path is formed by connecting three points $(1, 0)$, $(1, 1)$ and $(0, 1)$ with straight line segments (Fig. 1(a)). The corresponding joint path can be approximated by first converting the three Cartesian corner points to joint points and then connecting them with straight line segments in joint space (see Fig. 1(b)). Note that additional Cartesian intermediate knot points may be required to meet the prescribed accuracy in the transformation error. See Fig. 1(c) for one such example in which the middle points of the Cartesian path is chosen as an intermediate knot point. The knot point is then treated just like a regular Cartesian corner point (Fig. 1(d)). Although selection of intermediate knot points is an interesting research problem, we will not pursue it here since it is out of the scope of this paper.

Usually a joint path consists of three stages: acceleration, cruise at a constant velocity, and deceleration.[4] When transition from one path segment to another is to be made, it should be accomplished as fast as possible while meeting the tolerance requirement in the path deviation. When the

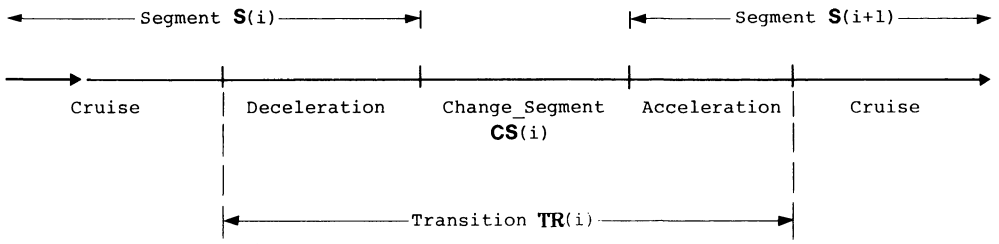

Fig. 2. Transition $TR(i)$ and change segment $CS(i)$.

robot arm moves along a first segment at its cruise speed, there are two possibilities for switching to the next segment, depending upon the tolerance of the path deviation. If the tolerance is tight, then the robot arm must first slow down before actual transition takes place. Otherwise, the robot arm can initiate the transition directly from its cruise speed along the first segment. Considering these two possibilities, we define the following two terms, namely, transition and change segment. Transition $TR(i)$ represents the stage of departing from the constant-velocity cruise along the segment $S(i)$ and arriving at the constant-velocity cruise along the next segment $S(i+1)$. Transition is desired to be made smoothly, without stopping, and within the specified path deviation. Change segment $CS(i)$ represents the stage of departing from $S(i)$ and then arriving at $S(i+1)$. See Fig. 2 for an illustration of $TR(i)$ and $CS(i)$. In general change segment $CS(i)$ is a part of transition $TR(i)$. The $TR(i)$ and $CS(i)$ would be the same if no acceleration or deceleration along the corresponding path segments is needed to satisfy the constraint on the path deviation at the corner point $p(i)$, but they would be different otherwise. If we allow large bounds for the path deviations at corner points, the total traveling time will be reduced due to the widened spatial freedom in robot motion. However, there will be an increase in the probability that the robot arm collides with obstacles. Hence, the path deviation bounds at corner points must be set by the task planner as a design variable, weighing between the total traveling time and the workspace requirement or collision avoidance.

It is well known that robot arm dynamics are highly nonlinear coupled functions of position, payload, mass, etc. Also, due to the joint actuator characteristics, there exist bounds on joint angular velocities and torques/forces [1].

Considering all the factors mentioned above, the global minimum-time path planning (GMTPP) problem in joint space can be stated as follows.

GMTPP Problem: Given a path composed of M segments $S(i)$, $i = 1, 2, \cdots, M$, formed by connecting $(M+1)$ corner points $p(i)$, $i = 0, 1, \cdots, M$ with straight line segments in joint space, find a minimum-time traveling path that the robot arm follows within the prescribed path deviation bounds at corner points $e(i)$, $i = 1, 2, \cdots, M-1$ with the initial position and velocity $q(0) = p(0)$, $\dot{q}(0) = 0$ and the final position and velocity $q(t_f) = p(M)$, $\dot{q}(t_f) = 0$ subject to the limits on joint angular velocities, $|v^j| \leqslant v^j_{\max}$, and bounds on joint torques/forces $|u^j| \leqslant u^j_{\max}$, $j = 1, 2, \cdots, n$, where n is the number of the robot arm joints.

The GMTPP problem naturally leads to a nonlinear programming problem with high dimensionality. If a traditional trapezoidal velocity profile (i.e., constant acceleration → cruise → constant deceleration) is assumed for each segment, there will be $3M$ unknowns for the entire path planning. As evidenced in [8], this problem becomes very difficult to solve even when robot arm dynamics and the absolute path deviation are not considered. We have sought simple and, to some extent, heuristic solutions to the problem under the following assumption.

A1) Each segment $S(i)$ $i = 1, \cdots, M$ is assumed to consist of three segments, namely *acceleration*, *cruise* with a maximum allowable velocity, and *deceleration* for transition from $S(i)$ to $S(i+1)$.

Assumption A1 is realistic for many robot applications, particularly for gross motion and minimum-time controls. On the other hand, the assumption may not be realistic when the robot undergoes many short moves, i.e., *fine motion planning*. However, when we are concerned with such short, fine, and acrobatic motions, there are many more important and compelling requirements to be met than the minimum-time motion. This implies that A1 is reasonable in the context of the minimum-time path planning, although it does not hold for a fine motion planning. Since A1 may become invalid for short moves not only in joint space but also in Cartesian space, use of A1 in [8] is believed to be justified by the same reasoning as described previously.

Furthermore, under A1 the GMTPP problem can be decomposed into a set of local minimum-time path-planning (LMTPP) problems. (In addition to the inclusion of the robot arm dynamics, this is a novel departure from [8], leading to a much simpler algorithm which can yet include the constraints on absolute path deviations.) Without loss of generality, for each LMTPP problem we can choose initial and final points $q_0(i)$ and $q_f(i)$ from $S(i)$ and $S(i+1)$, respectively, at which the robot arm is to attain the maximum allowable velocities $w_{\max}(i)$ and $w_{\max}(i+1)$, respectively (i.e., points located at their respective cruising portions of the path). The method for setting $q_0(i)$ and $q_f(i)$ will be discussed later in detail. The $w_{\max}(i)$, the maximum allowable velocity along $S(i)$, can be repre-

sented by

$$w_{\max}(i) = \zeta(i)r(i), \tag{1}$$

where $r(i)$ represents a unit vector along $S(i)$, i.e., $r(i) = S(i)/|S(i)| = (r^1(i), r^2(i), \cdots, r^n(i))^T$, and $\zeta(i)$ denotes the magnitude of the maximum allowable velocity in the direction of $r(i)$. Using the limits on joint angular velocity $v_{\max} = (v^1_{\max}, v^2_{\max}, \cdots, v^n_{\max})^T$, $\zeta(i)$ can be actually computed by

$$\zeta(i) = \min_j \frac{\sigma(r^j(i))v^j_{\max} + \sigma(-r^j(i))(-v^j_{\max})}{r^j(i)}, \tag{2}$$

where $\sigma(z)$ is defined by

$$\sigma(z) = \begin{cases} 1, & \text{if } z > 0 \\ 0, & \text{if } z \leqslant 0. \end{cases}$$

Then, the GMTPP problem is converted to a set of LMTPP(i) for $i = 1, 2, \cdots, M-1$ as follows.

LMTPP(i) Problem: Given a path composed of two segments $S(i)$ and $S(i+1)$, formed by connecting $q_0(i)$, $p(i)$, and $q_f(i)$ with straight line segments in joint space, find a minimum-time traveling path that the robot arm can follow within the prescribed deviation bound $e(i)$ at the corner point $p(i)$, subject to the limits on joint angular velocities $|v_j| \leqslant v^j_{\max}$ and bounds on joint torques/forces $|u_j| \leqslant u^j_{\max}$, $j = 1, \cdots, n$.

For the LMTPP problem, one may obtain necessary conditions for optimal solutions, resulting in a bang-bang solution in the nonsingular region. However, due to the coordination requirement among the joints of a single robot arm in following segments $S(i)$, $i = 1, \cdots, M$, control inputs for all but one joint (i.e., the slowest joint) are not to be bang-bang, meaning that there exist singular regions for all but one joint. Moreover, because of the complex nonlinear, coupled dynamics of robot arms, it is almost impossible to obtain any analytic or numerical solution to the LMTPP problem. Consequently, we explore some intrinsic properties of this problem and find a suboptimal solution with an additional assumption.

A2) A constant acceleration $a_c(i)$ is assumed during each change segment $CS(i)$, $i = 1, 2, \cdots, M$.

Assumption A2 is employed to simplify the analysis of the path deviations at corner points by utilizing the local acceleration bounds around corner points $p(i)$, $i = 1, 2, \cdots, M-1$. Note, however, that this assumption does not impose any unrealistic demand on the path planning; if A2 is deemed unrealistic, one can divide $CS(i)$ into a finite number of subregions within each of which a constant acceleration is then assumed. Although this refinement may improve transition timing at the expense of computational simplicity, we have not taken this course in Section V for clarity of our main purpose. Note that A2 was also previously used in [18], where joint positions during a transition were interpolated as a *quadratic* function of time.

III. Torque-to-Acceleration Conversion of Constraints

Since both the GMTPP and LMTPP problems are naturally related to accelerations rather than to torques/forces, it is necessary to convert the constraints on torques/forces to those on accelerations. For this conversion we consider the nonlinear, coupled robot arm dynamics. Using the Lagrangian formulation, the dynamics of the robot arm can be described by

$$D(q)\ddot{q} + h(q,\dot{q}) + g(q) = u, \tag{3}$$

where u is an $n \times 1$ generalized force/torque vector, and $q, \dot{q}, \ddot{q}$ are $n \times 1$ vectors of generalized coordinates, velocities, and accelerations, respectively. The $D(q)$ is an $n \times n$ inertial matrix, $h(q,\dot{q})$ is an $n \times 1$ viscous friction, Coriolis, and centrifugal force vector, $g(q)$ is an $n \times 1$ gravitational loading vector, and n is the number of joints of the robot arm. The inertia, gravity loading, and Coriolis and centrifugal terms depend on the position of each joint as well as on the mass, first moment, and inertia of each link. Also note that these terms are functions of the robot arm's payload (i.e., tool and parts).

The constraints on torques/forces are related to acceleration by

$$-u_{\max} \leqslant D(q)\ddot{q} + h(q,\dot{q}) + g(q) \leqslant u_{\max} \tag{4}$$

where $u_{\max} = (u^1_{\max}, u^2_{\max}, \cdots, u^n_{\max})^T$. If both q and $\dot{q}$ are known, then bounds on $\ddot{q}$ can be determined from the above inequality. However, these are unknown at the time of path planning, and hence some sort of approximations are needed. Since the constraints conversion is required only in the vicinity of the corner points,[5] such approximations can be made rather easily and realistically.

We have adopted an approximation algorithm for converting the bounds on joint torques/forces to those on joint accelerations around the corner points. The algorithm can be described as follows. At every corner point $p(i)$, $i = 1, 2, \cdots, M-1$ we compute the parameters of robot arm dynamics for three distinct cases with velocities (i.e., $\dot{q}$) $w_{\max}(i)$, $w_{\max}(i+1)$, and an average velocity $w_{\text{avg}}(i)$, but with the same position $q_c(i) = p(i) + e(i)$,[6] where $e(i)$ is defined as a path deviation vector around $p(i)$ with magnitude $e(i)$ and with the same direction as the vector $-r(i) + r(i+1)$. The $w_{\max}(i)$ is the cruise velocity approaching $TR(i)$, $w_{\max}(i+1)$ is the cruise velocity departing from $TR(i)$, and the average velocity $w_{\text{avg}}(i) = (w_{\max}(i) + w_{\max}(i+1))/2$ is used for approximating the velocity in the middle of change segment $CS(i)$. If it is desired to set a uniform acceleration bound around each

[5] We only need the local acceleration bounds in the transition stages $TR(i)$, $i = 1, 2, \cdots, M-1$, since maximum cruising velocity is applied outside transition stages.

[6] Approximate values of q and $\dot{q}$ can be calculated for as many points around the corner point $p(i)$ as necessary. Those values must be approximate on the basis of $w_{\max}(i)$, $w_{\max}(i+1)$, $p(i)$, and $e(i)$. For simplicity, we have used here only three different values of q and $\dot{q}$.

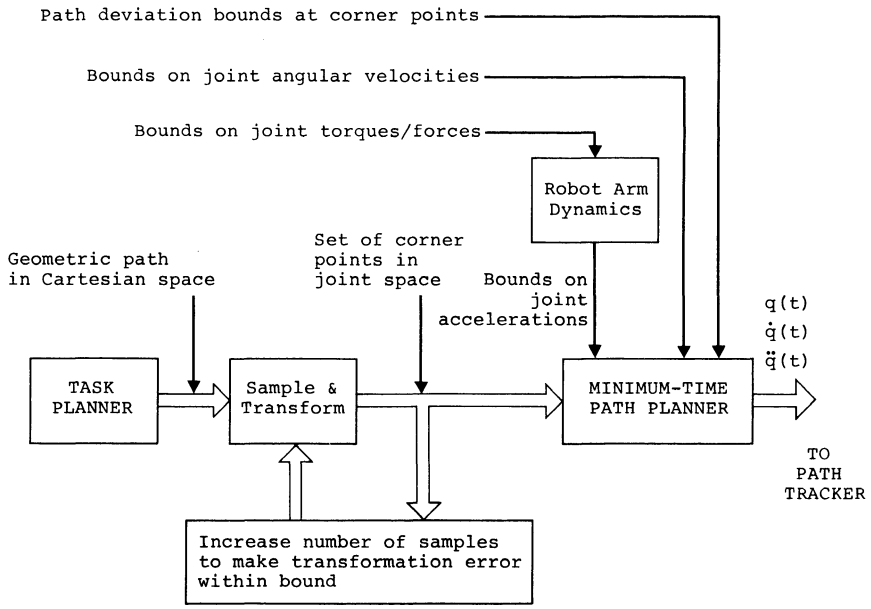

Fig. 3. Description of input and output of minimum-time path planner.

corner point $\boldsymbol{p}(i)$, one can select bounds $c^j_{\max}(i)$ and $c^j_{\min}(i)$ during transition $\boldsymbol{TR}(i)$. That is

$$c^j_{\min}(i) \leqslant \boldsymbol{D}^j(\boldsymbol{q}_c(i))\boldsymbol{a}(i) \leqslant c^j_{\max}(i), \qquad j = 1,2,\cdots,n, \tag{5}$$

where $\boldsymbol{a}(i)$ denotes the acceleration during $\boldsymbol{TR}(i)$, $\boldsymbol{D}^j$ the jth row of the matrix $\boldsymbol{D}$, and

$$\begin{aligned} c^j_{\max}(i) = \min \{ & u^j_{\max} - h^j(\boldsymbol{q}_c(i), \boldsymbol{w}_{\text{avg}}(i)) - g^j(\boldsymbol{q}_c(i)), \\ & u^j_{\max} - h^j(\boldsymbol{q}_c(i), \boldsymbol{w}_{\max}(i)) - g^j(\boldsymbol{q}_c(i)), \\ & u^j_{\max} - h^j(\boldsymbol{q}_c(i), \boldsymbol{w}_{\max}(i+1)) - g^j(\boldsymbol{q}_c(i)) \} \end{aligned} \tag{6a}$$

$$\begin{aligned} c^j_{\min}(i) = \max \{ & u^j_{\min} - h^j(\boldsymbol{q}_c(i), \boldsymbol{w}_{\text{avg}}(i)) - g^j(\boldsymbol{q}_c(i)), \\ & u^j_{\min} - h^j(\boldsymbol{q}_c(i), \boldsymbol{w}_{\max}(i)) - g^j(\boldsymbol{q}_c(i)), \\ & u^j_{\min} - h^j(\boldsymbol{q}_c(i), \boldsymbol{w}_{\max}(i+1)) - g^j(\boldsymbol{q}_c(i)) \}. \end{aligned} \tag{6b}$$

This results in a feasible region with a polyhedron boundary for valid accelerations during $\boldsymbol{TR}(i)$.

IV. The Minimum-Time Path Planning

The functional relationship of the proposed minimum-time path planner to other components in the system is described in Fig. 3. With the preceding assumptions and discussions, the LMTPP(i) problem can be solved by the following steps for $i = 1,\cdots,M-1$ (see Fig. 4 for an illustration of these steps).[7]

S1) Cruise with the maximum velocity $\boldsymbol{w}_0(i) = \boldsymbol{w}_{\max}(i)$ along the segment $\boldsymbol{S}(i)$ from $\boldsymbol{q}_0(i)$ to $\boldsymbol{q}_a(i)$, where $\boldsymbol{q}_a(i)$ is the point on $\boldsymbol{S}(i)$ at which $\boldsymbol{TR}(i)$ begins.

[7]Both start and termination of motion are not included here but can be found in the entire path-planning algorithm near the end of this section.

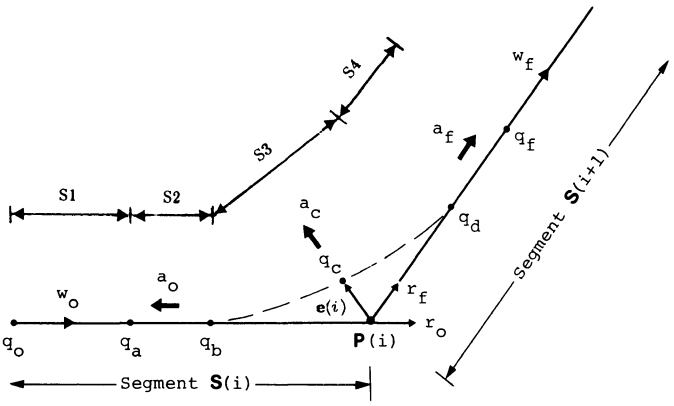

Fig. 4. Individual steps in transition $\boldsymbol{TR}(i)$.

S2) Apply a constant deceleration $\boldsymbol{a}_0(i)$ along segment $\boldsymbol{S}(i)$ from $\boldsymbol{q}_a(i)$ to $\boldsymbol{q}_b(i)$ at which $\boldsymbol{CS}(i)$ begins.

S3) Change segments (from $\boldsymbol{S}(i)$ to $\boldsymbol{S}(i+1)$) with constant acceleration $\boldsymbol{a}_c(i)$ from $\boldsymbol{q}_b(i)$ to $\boldsymbol{q}_d(i)$ passing through $\boldsymbol{q}_c(i)$, where $\boldsymbol{q}_c(i) = \boldsymbol{p}(i) + \boldsymbol{e}(i)$, and $\boldsymbol{q}_d(i)$ is the point at which $\boldsymbol{CS}(i)$ terminates.

S4) Apply a constant acceleration $\boldsymbol{a}_f(i)$ along segment $\boldsymbol{S}(i+1)$ from $\boldsymbol{q}_a(i)$ to $\boldsymbol{q}_f(i)$ attaining the cruise velocity $\boldsymbol{w}_f(i) = \boldsymbol{w}_{\max}(i+1)$ on $\boldsymbol{S}(i+1)$. Note that $\boldsymbol{TR}(i)$ terminates at $\boldsymbol{q}_f(i)$.

Steps S2 and S4 guarantee the solution even if $e(i)$ is so small or tight that the direct transition from $\boldsymbol{w}_0(i)$ to $\boldsymbol{w}_f(i)$ with a constant acceleration $\boldsymbol{a}_c(i)$ cannot be achieved. Hence, three constant accelerations are used here, namely, decelerating with $\boldsymbol{a}_0(i)$ along $\boldsymbol{S}(i)$, changing segments with $\boldsymbol{a}_c(i)$ from $\boldsymbol{S}(i)$ to $\boldsymbol{S}(i+1)$, and then accelerating with $\boldsymbol{a}_f(i)$ along $\boldsymbol{S}(i+1)$. These three accelerations are to be determined by the present path planner.

Define the unit vectors along $\boldsymbol{S}(i)$ and $\boldsymbol{S}(i+1)$, respectively, as

$$\boldsymbol{r}_0(i) = \boldsymbol{r}(i); \qquad \boldsymbol{r}_f(i) = \boldsymbol{r}(i+1). \tag{7}$$

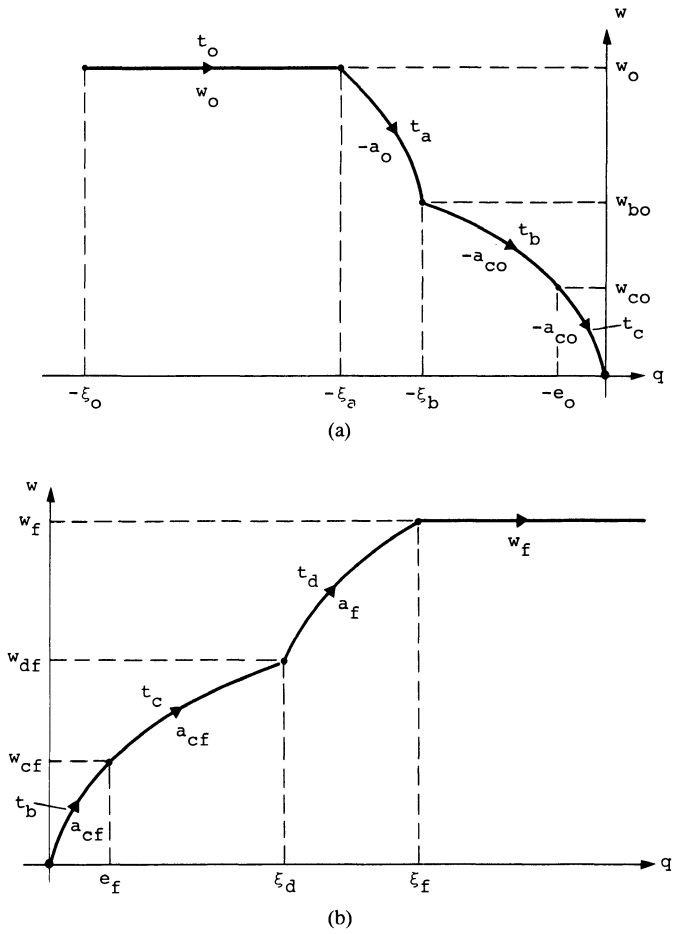

Fig. 5. Transition components in state space. (a) r_0 component. (b) r_f component.

Then every vector can be uniquely decomposed into $r_0(i)$ and $r_f(i)$ components as follows (see Fig. 4)

$$\begin{aligned} e(i) &= -e_0(i)r_0(i) + e_f(i)r_f(i) \\ a_0(i) &= -a_0(i)r_0(i) \\ a_f(i) &= a_f(i)r_f(i) \\ a_c(i) &= -a_{c0}(i)r_0(i) + a_{cf}(i)r_f(i) \\ w_0(i) &= w_0(i)r_0(i) \\ w_f(i) &= w_f(i)r_f(i) \\ p(i) - q_0(i) &= \xi_0(i)r_0(i) \\ q_f(i) - p(i) &= \xi_f(i)r_f(i) \end{aligned} \tag{8}$$

where scalar quantities with subscripts 0 and f represent their respective r_0 and r_f components. The $\xi_0(i)$ and $\xi_f(i)$ denote the magnitudes of vectors $p(i) - q_0(i)$ and $q_f(i) - p(i)$, respectively. One can plot each transition component in state space as depicted in Fig. 5. In Step S4, the final condition $q_f(i)$ is set as a position when the velocity reaches $w_{\max}(i+1)$ with constant acceleration $a_f(i)$. This value of $q_f(i)$ is used as the initial position in the next segment, i.e., $q_0(i+1) = q_f(i)$. Depending upon whether Steps S2 and S4 are required or not, we can consider four cases in the following analysis. Using the state space trajectory in Fig. 5, we can compute the traveling time $T(i)$ required to move from $q_0(i)$ to $q_f(i)$ and positions $q_a(i)$, $q_b(i)$, and $q_d(i)$, as functions of $a_0(i)$, $a_{c0}(i)$, $a_{cf}(i)$, $a_f(i)$, $w_0(i)$, and $w_f(i)$. The dependence on segment i is omitted for notational simplicity both in Fig. 5 and in the following discussion. For convenience, let

$$\tau \equiv \sqrt{\frac{2e_0}{a_{c0}}} + \sqrt{\frac{2e_f}{a_{cf}}} \quad \text{and} \quad c_0 \equiv \frac{w_0}{2a_0} + \frac{w_f}{2a_f} + \frac{\xi_0}{w_0} + \frac{\xi_f}{w_f}.$$

1) When $a_{c0}\tau < w_0$, $a_{cf}\tau < w_f$, i.e., cruise → deceleration → change segment → acceleration → cruise,

$$q_a(i) = p - \left[\frac{1}{2}a_{c0}\left(1 - \frac{a_{c0}}{a_0}\right)\tau^2 + \frac{w_0^2}{2a_0}\right]r_0$$

$$q_b(i) = p - \frac{1}{2}a_{c0}\tau^2 r_0$$

$$q_d(i) = p + \frac{1}{2}a_{cf}\tau^2 r_f$$

$$q_f(i) = p + \left[\frac{1}{2}a_{cf}\left(1 - \frac{a_{cf}}{a_f}\right)\tau^2 + \frac{w_f^2}{2a_f}\right]r_f$$

$$T(i) = c_0 + r\left(1 - \frac{a_{c0}}{a_0} - \frac{a_{cf}}{a_f}\right) - \tau^2\left[\frac{a_{c0}}{2w_0}\left(1 - \frac{a_{c0}}{a_0}\right) + \frac{a_{cf}}{2w_f}\left(1 - \frac{a_{cf}}{a_f}\right)\right]. \tag{9a}$$

2) When $a_{c0}\tau \geq w_0$, $a_{cf}\tau < w_f$, i.e., cruise → change segment → acceleration → cruise,

$$q_0(i) = q_a(i) = p - \frac{w_0^2}{2a_{c0}}r_0$$

$$q_d(i) = p + \frac{1}{2}a_{cf}\left(\frac{w_0}{a_{c0}}\right)^2 r_f$$

$$q_f(i) = p + \left[\frac{1}{2}a_{cf}\left(1 - \frac{a_{cf}}{a_f}\right)\left(\frac{w_0}{a_{c0}}\right)^2 + \frac{w_f^2}{2a_f}\right]r_f$$

$$T(i) = c_0 - \frac{w_0}{a_{c0}}\frac{a_{cf}}{a_f} - \left(\frac{w_0}{a_{c0}}\right)^2\left(1 - \frac{a_{cf}}{a_f}\right)\frac{a_{cf}}{2w_f}. \tag{9b}$$

3) When $a_{c0}\tau < w_0$, $a_{cf}\tau \geq w_f$, i.e., cruise → deceleration → change segment → cruise,

$$q_a(i) = p - \left[\frac{1}{2}a_{c0}\left(1 - \frac{a_{c0}}{a_0}\right)\left(\frac{w_f}{a_{cf}}\right)^2 + \frac{w_0^2}{2a_0}\right]r_0$$

$$q_b(i) = p - \frac{1}{2}a_{c0}\left(\frac{w_f}{a_{cf}}\right)^2 r_0$$

$$q_d(i) = q_f(i) = p - \frac{w_f^2}{2}a_{cf}r_0$$

$$T(i) = c_0 - \frac{w_f}{a_{cf}}\frac{a_{c0}}{a_0} - \left(\frac{w_f}{a_{cf}}\right)^2\left(1 - \frac{a_{cf}}{a_0}\right)\frac{a_{cf}}{2w_0}. \tag{9c}$$

4) When $a_{c0}\tau \geq w_0$, $a_{cf}\tau \geq w_f$, i.e., cruise → change segment → cruise,

$$\boldsymbol{q}_a(i) = \boldsymbol{q}_b(i) = \boldsymbol{p} - \frac{w_0^2}{2a_{c0}}\boldsymbol{r}_0$$

$$\boldsymbol{q}_d(i) = \boldsymbol{q}_f(i) = \boldsymbol{p} + \frac{w_f^2}{2\boldsymbol{a}_{cf}}\boldsymbol{r}_f$$

$$T(i) = \frac{\xi_0}{w_0} + \frac{\xi_f}{w_f}. \tag{9d}$$

Now, consider the calculation of $\boldsymbol{a}_0(i)$ and $\boldsymbol{a}_f(i)$. We can easily show that $\partial T(i)/\partial a_0(i) < 0$, $i = 1, \cdots, M-1$ for the cases 1) and 3). This property implies that $a_0(i)$ should take as large a value as possible. Likewise, $a_f(i)$ should take as large a value as possible. Hence, we can set the values of $\boldsymbol{a}_0(i)$ and $\boldsymbol{a}_f(i)$ to attain maximum magnitudes in the directions of $-\boldsymbol{r}_0(i)$ and $\boldsymbol{r}_f(i)$, respectively.

To determine $\boldsymbol{a}_c(i)$, which consists of two components $a_{c0}(i)$ and $a_{cf}(i)$ we have to use the polyhedron boundary of the accelerations given by (5). Then the problem becomes a two-dimensional nonlinear optimization problem of minimizing (9) with respect to $a_{c0}(i)$ and $a_{cf}(i)$ subject to linear constraints (5). For a solution to this problem, we can get the following useful property.

Property: $T(i)$ attains its minimum when the components of $\boldsymbol{a}_c(i)$ are maximized within the feasible region.

It is easy to show $\partial T(i)/\partial a_{c0}(i) \leqslant 0$ and $\partial T(i)/\partial a_{cf}(i) \leqslant 0$. With these inequalities, proof of the above property is straightforward. The inequalities imply that for the minimum of $T(i)$ the components of the acceleration $\boldsymbol{a}_c(i)$ should be maximized. But the values of $a_{c0}(i)$ and $a_{cf}(i)$ are interrelated via the linear constraints (5). Hence the minimum of $T(i)$ occurs at the boundary of inequalities (5), resulting in a one-dimensional optimization problem.

The acceleration during change segment $\boldsymbol{CS}(i)$, $\boldsymbol{a}_c(i)$, can be obtained with a suitable bisection algorithm searching along the boundary of (5). Let θ be the angle between $\boldsymbol{a}_0(i)$ and $\boldsymbol{a}_f(i)$. The following iterations are performed for $k = 1, 2, \cdots$. Divide the angle θ and generate a set of accelerations $\boldsymbol{a}_l$ having angles $[(2l-1)/2^k]\theta$, $l = 1, 2, \cdots, 2^{k-1}$, and the corresponding traveling times T_{kl}. Compare T_{kl} and choose the minimum (which occurs at $l = l'$) and set it to T_k. If the bisecting angle $2^{-k}\theta$ gets smaller and improvement in T_k from T_{k-1} becomes insignificant, then the algorithm terminates with $\boldsymbol{a}_c(i) = \boldsymbol{a}_{l'}$.

For (the first) segment $\boldsymbol{S}(1)$, we need a maximum acceleration until the velocity reaches $\boldsymbol{w}_{\max}(1)$. No iteration is necessary for this segment. We can simply compute the maximum acceleration, and set $\boldsymbol{q}_f(0) = \boldsymbol{q}_0(1)$ for the next segment. For (the last) segment $\boldsymbol{S}(M)$, we need to cruise with $\boldsymbol{w}_{\max}(M)$ followed by the maximum deceleration to the final position $\boldsymbol{p}(M)$ and zero velocity. This segment can be considered as a reverse procedure of the first segment. The maximum deceleration time is computed backwards in time.

The minimum-time path-planning algorithm discussed thus far can be summarized by the following algorithm:

1) Set $i = 0$.
   1.1) Compute the maximum acceleration $\boldsymbol{a}_f(0)$.
   1.2) Compute the traveling time $T(0)$.
   1.3) Compute the initial position for $\boldsymbol{q}_0(1)$ $\boldsymbol{S}(1)$.
2) Set $i = i+1$, then
   2.1) Compute the unit vectors $\boldsymbol{r}_0(i)$ and $\boldsymbol{r}_f(i)$.
   2.2) Compute a position $\boldsymbol{q}_c(i)$ with the maximum path deviation during transition.
   2.3) Compute velocity bounds for $\boldsymbol{w}_0(i)$ and $\boldsymbol{w}_f(i)$.
   2.4) Compute the maximum deceleration $\boldsymbol{a}_0(i)$ along $\boldsymbol{S}(i)$ and the maximum acceleration $\boldsymbol{a}_f(i)$ along $\boldsymbol{S}(i+1)$.
   2.5) Compute the acceleration $\boldsymbol{a}_c(i)$ during change segment $\boldsymbol{CS}(i)$, and the traveling time $T(i)$ by using the bisecting algorithm.
   2.6) Compute the initial position $\boldsymbol{q}_0(i+1)$ of the next segment.
   2.7) If $i < M$, then go to Step 2.
3) For the last segment (i.e., $i = M$)
   3.1) Compute the maximum deceleration $\boldsymbol{a}_0(M)$ toward $(\boldsymbol{p}(M), 0)$.
   3.2) Compute the traveling time $T(M)$.
4) Compute the total traveling time $T_{\text{total}} = \sum_{i=0}^{M} T(i)$.

As can be seen in the above, the present path-planning algorithm requires only rudimentary calculations and simple one-dimensional optimizations; this is in sharp contrast to the method developed in [8] where a mathematical programming problem was solved for multiple variables by a complex approximate optimization technique.

V. A Path-Planning Example

Using a simulator of the PUMA 600 robot arm on a DEC VAX 11/780, we have simulated the proposed minimum-time path-planning algorithm. The PUMA arm is manufactured by Unimation, Inc., and consists of six rotational joints, each of which is driven by a dc servomotor.

We employed the Lagrangian formulation to derive the PUMA arm dynamics as in (10) with the coordinate system using the Denavit and Hartenberg representation in Table I, which is then used to simulate the behavior of the first three joints of the arm. The remaining three joints are not considered here for simplicity

$$\sum_{j=1}^{n} d_{ij}(\boldsymbol{q})\ddot{q}_j + \sum_{j=1}^{n}\sum_{k=1}^{n} h_{ijk}(\boldsymbol{q})\dot{q}_j\dot{q}_k + g_i(\boldsymbol{q}) = u_i, \quad i = 1, 2, \cdots, n. \tag{10}$$

Typical terms have the following form:

$$\begin{aligned} d_{11}(q) &= J_{111} + J_{133} + J_{233} + 2J_{234}d_2 + J_{244}d_2^2 \\ &\quad + J_{322} + J_{333} + J_{334}d_2^2 \\ &\quad + \left(J_{211} + 2J_{214}a_2 + J_{244}a_2^2 + J_{344}a_2^2\right)C_2^2 + J_{222}S_2^2 \\ &\quad + (J_{311} - J_{333})C_{23}^2 + 2J_{334}a_2C_2S_{23} \\ h_{211} &= \left(J_{211} - J_{222} + 2J_{214}a_2 + J_{244}S_2 + J_{344}a_2^2\right)S_2C_2 \\ &\quad + (J_{311} - J_{333})C_{23}S_{23} - J_{334}a_2(C_2C_{23} - S_2S_{23}) \\ g_2 &= -(a_2m_3 + a_2m_2 + m_2\bar{x}_2)C_2g - m_3\bar{z}_3S_{23}g, \end{aligned}$$

where J_{ijk} is the (i,k)th element of the 4×4 inertia

TABLE I
LINK COORDINATE SYSTEM FOR A PUMA 600 ROBOT

Joint i	θ_i	d_i (mm)	a_i (mm)	α'_i (deg.)	Range (deg.)
1	θ_1	0	0	-90	-160 to 160
2	θ_2	149.5	432	0	-225 to 45
3	θ_3	0	0	90	-45 to 225

TABLE II
MASS, FIRST MOMENTS, AND INERTIAS OF THE FIRST THREE JOINTS FOR THE PUMA 600 MANIPULATOR

Link	Mass	Center of Mass			Inertia		
	M (Kg)	$\bar{x}$ (m)	$\bar{y}$ (m)	$\bar{z}$ (m)	I_x ($Kg\ m^2$)	I_y ($Kg\ m^2$)	I_z ($Kg\ m^2$)
1	2.27	0.0	0.0	0.075	0.00376	0.00376	0.0169
2	15.91	-0.216	0.0	0.0	0.9897	0.1237	0.1237
3	11.36	0.0	0.0	0.216	0.0074	0.0074	0.7067

TABLE III
CORNER POINTS AND TOLERANCES IN PATH DEVIATION USED FOR SIMULATION

i	Corner points, p(i)			Tolerances
	Joint 1 (deg.)	Joint 2 (deg.)	Joint 3 (deg.)	e(i) (deg.)
0	0.0	0.0	90.0	0.0
1	0.0	-90.0	135.0	1.0
2	90.0	-90.0	135.0	1.0
3	90.0	0.0	90.0	1.0
4	0.0	0.0	90.0	0.0

tensor, J_i, for the ith joint; d_2, a_2 are lengths related to the arm coordjnate frame; $(\bar{x}_i, \bar{y}_i, \bar{z}_i)$ is the center of mass for link i; and for $i, j = 1,2,3$ $C_i = \cos(q_i)$, $S_i = \sin(q_i)$, $C_{ij} = \cos(q_i + q_j)$, $S_{ij} = \sin(q_i + q_j)$.

For the PUMA simulator the above dynamic equations are computed with the numerical values of the mass, center of mass, and inertia for each joint given in Table II. These are approximate figures acquired from the manufacturer's specification.

As shown in Table III, we selected a set of corner points $p(i)$, $i = 0, 1, \cdots, M$, and path deviation bounds at these corner points $e(i)$, $i = 1, \cdots, M-1$. The bounds on the

TABLE IV
TRAVELING TIME

	T_{cruise} (sec.)	$T_{trans.}$ (sec.)	T_{total} (sec.)
Local bounds	3.015	1.733	4.748
Global bound	1.320	4.860	6.180

TABLE V
EFFECTS OF PATH DEVIATION BOUNDS ON TOTAL TRAVELING TIME

Deviation bound (deg.)	T_{cruise} (sec.)	$T_{trans.}$ (sec.)	T_{total} (sec.)
0.0	3.024	1.951	4.976
1.0	3.015	1.733	4.748
2.0	3.009	1.714	4.723
4.0	3.006	1.704	4.710

control input torques are assumed to be $|u_1| \leqslant 100$ N · m, $|u_2| \leqslant 150$ N · m, and $|u_3| \leqslant 50$ N · m, and the maximum angular velocity is set to 90°/s for each joint. Observe that the choice of these simulation values is arbitrary for the sake of numerical demonstration, and any of such choices does not change the basic performance of our path-planning method. The path planner computes the desired path (i.e., $\boldsymbol{q}$ and $\dot{\boldsymbol{q}}$) which requires the minimum total traveling time, using the algorithm developed in the previous sections. In order to examine the performance of the minimum-time path planner we compared it with a path-planning method with global bounds on the acceleration with the same simulation data.[8] The corresponding simulation results are given in Table IV. It shows that the minimum-time path planner developed in this paper exhibits excellent transition characteristics when compared with the one with global acceleration bounds. For this particular example, the transition time T_{trans}, is reduced to about one third of that with the global acceleration bound. The cruise time with local acceleration bounds is somewhat longer than the case with a global acceleration bound, because the fast transition needs shorter transition distance and hence longer cruising distance. Including both the transition and cruise characteristics for the example, the present path-planning method showed a significant improvement in the total traveling time T_{total} (a 23% reduction).

The effects of the path deviation bound at each corner point are simulated and presented in Table V. Here, all $e(i)$ are set to an equal value for $i = 1, \cdots, M-1$. As expected, the larger the deviation bound, the less the total traveling time because of the spatial freedom in motion. Particularly, one can see a drastic improvement in T_{trans} at the beginning and then a slow improvement or a near saturation as the deviation tolerances increase. Note, however, that T_{cruise} is relatively insensitive to the magnitudes of $e(i)$, $i = 1, 2, \cdots, M$.

As a whole, for the example considered here our minimum-time path planning has indeed shown a significant improvement in the total traveling time.

VI. CONCLUDING REMARKS

We have developed a minimum-time path-planning method in joint space with robot arm dynamics included. An absolute path deviation bound for each corner point

[8] Obviously, the path solution with global acceleration bounds is *different* from the solution to the GMTPP problem.

can be specified as a design variable. Local upper bounds on joint accelerations are derived from robot arm dynamics so as to nearly fully utilize the robot's capabilities, and a set of local one-dimensional optimization problems can replace the global minimum-time problem. The local optimization problem is computationally simple with a small number of variables for each pair of segments, whereas the global optimization problem is computationally demanding since 1) a large number of variables should be considered simultaneously, and 2) special optimization techniques are required to get approximate solutions. Furthermore, for the GMTPP problem the number of variables in the optimization process varies with the number of segments that form the entire path, whereas it does not for the LMTPP problem. The simulation results for the present method show that the transition times are considerably improved, leading to a significant reduction in the total traveling time.

The torque-to-acceleration conversion of constraints was made on the basis of a heuristic approximation rather than an exact solution to the dynamic equations for the acceleration. It is impossible to obtain the exact solution, yet the approximation can be controlled to provide realistic conversion accuracy (by calculating acceleration bounds for as many subregions around a corner point as necessary). One important assumption in this conversion is that the exact dynamic equations are known. Generally this assumption is not valid. It is, however, a realistic assumption in view of the fact that the robot arm is to execute the same task repetitively many times and, therefore, its dynamics can be learned prior to the actual execution (and also prior to path planning).

There is another interesting approach to the minimum-time path-planning problem, which was developed independently by both Bobrow *et al.* [17] and Shin and McKay [14]. Both used a parametric function to describe the desired geometric path and also employed the phase plane approach to solve the minimum-time problem, resulting in a sequence of alternating acceleration and deceleration. Although the basic ideas for these two works are the same, they are quite different in several respects, e.g., search algorithms for switching points, guaranteed convergence, nature of the feasible regions in the phase plane, etc. However, despite its elegance, the phase plane approach is not applicable to the problems with constraints either on jerk as in [8] or on the path deviations at corner points, as we discussed in this paper.

As one referee pointed out, due to practical reasons, the minimum-time solution may not be of the bang-bang type even with respect to the slowest joint actuator. However, once the robot arm dynamics are modeled by (3) and the controls are bounded, the minimum-time solution becomes the bang-bang type. This should be interpreted as a result of mathematical formalism rather than reality. For practical use of the bang-bang solution one may, for example, set bounds on the torques/forces tighter than the actual values.

Path planning has not received much attention despite its importance. Particularly, there are numerous publications in the area of path control or tracking, where a perfect or at least a good path planner is always assumed to exist. Contrary to this assumption, there have been only a few published results in path planning. The imbalance between the two has to be resolved for the optimal or near-optimal utilization of robots' capabilities. Consequently, a balanced combination of path planning and robot control is essential for future automation with robots. The minimum-time path planning presented here has aimed at this objective.

References

[1] R. L. McIntyre, *Electric Motor Control Fundamentals*, 3rd ed. New York: McGraw-Hill, 1974.

[2] M. Takegaki and S. Arimoto, "A new feedback method for dynamic control of manipulators," *Trans. ASME J. Dynamic Syst., Meas., Contr.*, vol. 103, pp. 119–125, June 1981.

[3] M. E. Kahn and B. E. Roth, "The near-minimum-time control of open-loop articulated kinematic chains," *Trans. ASME J. Dynamic Syst., Meas., Contr.*, vol. 93, no. 3, pp. 164–172, Sep. 1971.

[4] B. K. Kim and K. G. Shin, "Suboptimal control of industrial manipulators with a weighted minimum time-fuel criterion," in *Proc. 22nd Conf. Decision and Control*, San Antonio, TX, Dec. 1983, pp. 1199–1204. Also in *IEEE Trans. Automat. Contr.*, vol. AC-30, no. 1, pp. 1–10, Jan. 1985.

[5] J. Y. S. Luh, M. W. Walker, and R. P. C. Paul, "Resolved-acceleration control of mechanical manipulators," *IEEE Trans. Automat. Contr.*, vol. AC-25, no. 3, pp. 468–474, June 1980.

[6] ——, "On-line computational scheme for mechanical manipulators," *Trans. ASME J. Dynamic Syst., Meas., Contr.*, vol. 102, pp. 69–76, June 1980.

[7] R. Paul, "Manipulator Cartesian path control," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-9, pp. 702–711, Nov. 1979.

[8] 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.

[9] R. Paul, "The mathematics of computer controlled manipulator," in *Proc. Joint Auto. Contr. Conf.*, vol. 1, 1977, pp. 124–131.

[10] C. S. Lin, P. R. Chang, and J. Y. S. Luh, "Formulation and optimization of cubic polynomial joint trajectories for mechanical manipulators," in *IEEE Conf. Decision and Control*, Orlando, FL, Dec. 1982. Also in *IEEE Trans. Automat. Contr.*, vol. AC-28, no. 12, pp. 1066–1074, Dec. 1983.

[11] R. H. Taylor, "Planning and execution of straight line manipulator trajectories," *IBM J. Res., Develop.*, vol. 23, pp. 424–436, July 1979.

[12] J. Y. S. Luh, "An anatomy of industrial robots and their controls," *IEEE Trans. Automat. Contr.*, vol. AC-28, no. 2, pp. 133–153, Feb. 1983.

[13] T. Lozano-Perez, "Spatial planning: A configuration space approach," *IEEE Trans. Comput.*, vol. C-32, no. 2, pp. 108–119, Feb. 1983.

[14] K. G. Shin and N. D. McKay, "An efficient manipulator control under geometric path constraints," in *Proc. 22nd Conf. Decision and Control*, San Antonio, TX, Dec. 1983, pp. 1449–1457. Also in *IEEE Trans. Automat. Contr.*, vol. AC-30, no. 6, to be published.

[15] R. Featherstone, "Position and velocity transformations between robot end-effector coordinates and joint angles," *Int. J. Robotics Research*, vol. 2, no. 2, pp. 35–45, 1983.

[16] J. Hollerbach, "Dynamic scaling of manipulator trajectories," in *Proc. Amer. Control Conf.*, San Francisco, June 1983, pp. 752–756.

[17] J. E. Bobrow, S. Dubowsky, and J. S. Gibson, "On the optimal control of robotic manipulators with actuator constraints," in *Proc. Amer. Contr. Conf.*, San Francisco, June 1983, pp. 782–787.

[18] C. Rosen *et al.*, "Machine intelligence research applied to industrial automation," NSF Grant APR75-13074, SRI Int., Menlo Park, CA, no. 6 pp. 45–74, Nov. 1976.