WP2 - 3:30

AN EFFICIENT MINIMUM-TIME ROBOT PATH PLANNING UNDER REALISTIC CONDITIONS[1]

Byung Kook Kim and Kang G. Shin

Department of Electrical and Computer Engineering
The University of Michigan
Ann Arbor, Michigan 48109

ABSTRACT

In this paper, we have developed a method for minimum-time path planning in joint space subject to realistic constraints. This method differs from others in that: (i) an *absolute path deviation* at each corner point of the path can be specified, (ii) local upper bounds on joint accelerations are *derived from the arm dynamics* so as to nearly fully utilize robot's capabilities, and (iii) 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.

As a demonstrative example, we have applied the method to the path planning of the first three joints of the Unimation PUMA 600 series manipulator. The example has indeed shown significant improvements in the total traveling time in addition to the computational simplicity obtained from the decomposition of the global problem into a set of local problems.

INTRODUCTION

Industrial robots, more specifically industrial manipulators, have been regarded as one of the primary devices to be used for automated manufacturing due to their potential capabilities of material handling and assembly with speed, precision, and flexibility. The optimal control of manipulators is thus a key to the success of automated manufacturing. However, the optimal control of manipulators is known to be a very difficult problem due to (i) their nonlinear and coupled dynamics, (ii) physical constraints such as limits on the angular velocity and on the torque/force for each joint actuator[1], and (iii) possible collision with other objects in the workspace. The development of the optimal control of manipulators has been significantly hampered by these difficulties, and, therefore, there are only a few known attempts made, leading to suboptimal control solutions with performance indices of linear quadratic functional[2], minimum-time[3], and weighted time-fuel[4].

An alternative approach to the manipulator control problem is 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 manipulator dynamics. In general, path tracking takes manipulator 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 manipulator dynamics. This fact may be justified in view of the difficulty in obtaining path solutions, but the resulting timing of positions and velocities may force the robot to be under-utilized, 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,[2] and the constraints are given on the torque/force and angular velocity of each 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[12].

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 velocities and accelerations. In the Cartesian space path planning, the motion on the trajectory segment is well defined in the context of application, but its computational demand is high due to the required transformation of Cartesian coordinate points to joint coordinate points, and also becomes difficult to handle the case when the manipulator is at degenerate positions.

Due to its direct implementability, the path planning in joint space is also attractive. Paul[9] suggested a simple approach eliminating stopping at each transition from one path 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 joint. Taylor[11] proposed an iterative algorithm to compute sufficient number of intermediate points in joint space so that the transformation error -- difference between the original Cartesian path and the Cartesian path generated by

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

[1] *The work reported here is supported in part by the US AFOSR Contract No. F49620-82-C-0089 and Robot Systems Division, Center for Robotics and Integrated Manufacturing (CRIM), The University of Michigan, Ann Arbor, Michigan.*

Authorized licensed use limited to: University of Michigan Library. Downloaded on January 19,2023 at 21:13:41 UTC from IEEE Xplore. Restrictions apply.

transforming the interpolated joint paths back to the Cartesian space -- may become smaller than the prescribed value. The path planning in joint space is simple and fast, since it is limited only by maximum joint torques and velocities, and since degeneracies of the manipulator do not present any difficulty. It is particularly useful for specifying gross motion of the manipulator when (i) it operates in a collision-free space and (ii) 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 of all operating accelerations so as to enable the manipulator to follow the prespecified path under any operating condition (e.g. configuration and payload). It implies that the full capability of the manipulator cannot be utilized if the conventional approach is taken. Furthermore, the path deviations allowed at corner points are not considered explicitly. This is for the sake of simplicity of the solution rather than for the application reality.

In this paper, we propose a path-planning method in joint space by minimizing the total traveling time, given a set of corner points in joint space and realistic constraints, that is, those on the generated torques and angular velocity, and on the deviation at each corner point in the joint paths. Note that (i) limits on the generated torques/forces in place of accelerations introduce manipulator dynamics into path planning and (ii) 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). Moreover, the path planning to be developed here can be decomposed into a set of local optimizations under a certain condition.

This paper is organized as follows. In Section 2, the minimum time path planning problem is defined. Torque to acceleration conversion of constraints is discussed in Section 3, and a solution to the path planning problem is derived in Section 4. The proposed path planning is simulated for the first three joints of the PUMA 600 series manipulator in Section 5, showing the improvements in the total traveling time and the computational simplicity. The paper concludes with Section 6.

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 manipulator 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 $\mathbf{p}(i)$, $i = 0,1,\ldots,M$, in joint space can be obtained by transforming the Cartesian corner points (i.e. output of the geometric path planner) and possibly 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 prespecified accuracy is supplied by the task planner.

A path segment in joint space is formed by connecting two adjacent corner points with a straight line,[3] and typically consists of three stages: *acceleration*, *cruise* at a constant velocity, and *deceleration*. 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 manipulator 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 manipulator must first slow down before actual transition takes place. Otherwise, the manipulator 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 $\mathbf{TR}(i)$ represents the stage of departing from the constant velocity cruise along the segment $\mathbf{S}(i)$ and arriving at the constant velocity cruise along the next segment $\mathbf{S}(i+1)$. Transition is desired to be made smoothly, without stopping, and within the specified path deviation. Change_segment $\mathbf{CS}(i)$ represents the stage of departing from $\mathbf{S}(i)$ and arriving at $\mathbf{S}(i+1)$. See Fig. 1 for an illustration of $\mathbf{TR}(i)$ and $\mathbf{CS}(i)$. In general change_segment $\mathbf{CS}(i)$ is a part of transition $\mathbf{TR}(i)$; $\mathbf{TR}(i)$ and $\mathbf{CS}(i)$ 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 $\mathbf{p}(i)$, 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 manipulator motion. However, there will be an increase in the probability that the manipulator 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 manipulator 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.

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

Problem GMTPP naturally leads to a nonlinear programming problem with high dimensionality. If a traditional trapezoidal velocity profile (i.e. acceleration - cruise - 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 manipulator 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 $\mathbf{S}(i)$ $i = 1,\ldots,M$ is assumed to be large enough for the manipulator to accelerate, cruise with maximum allowable velocity, and decelerate for transition from $\mathbf{S}(i)$ to $\mathbf{S}(i+1)$ $i = 1,\ldots,M-1$.

[3] *At a first glance it may look strange to consider a "straight line segment" path in joint space. However, a moment of reflection and References [9],[10] will clarify this.*

Authorized licensed use limited to: University of Michigan Library. Downloaded on January 19,2023 at 21:13:41 UTC from IEEE Xplore. Restrictions apply.

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 manipulator undergoes many short moves. However, when we are concerned with such short, fine moves, there are many more important 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 is not in general. Note that the same assumption is applicable to Cartesian paths and hence was used in [8]. Furthermore, under A1 the GMTPP problem can be decomposed into a set of local minimum-time path planning(LMTPP) problems. Without loss of generality, for each LMTPP problem we can choose initial and final points $\mathbf{q}_o(i)$ and $\mathbf{q}_f(i)$ from $\mathbf{S}(i)$ and $\mathbf{S}(i+1)$, respectively, at which the manipulator is to attain the maximum allowable velocities, $\mathbf{w}_{\max}(i)$ and $\mathbf{w}_{\max}(i+1)$, respectively (i.e. points located at their respective cruising portions of the path). The method for setting $\mathbf{q}_o(i)$ and $\mathbf{q}_f(i)$ will be discussed later in detail. $\mathbf{w}_{\max}(i)$, the maximum allowable velocity along $\mathbf{S}(i)$, can be represented by

$$\mathbf{w}_{\max}(i) = \zeta(i)\,\mathbf{r}(i) \tag{1}$$

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

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

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

Problem LMTPP(i): Given a path composed of two segments $\mathbf{S}(i)$ and $\mathbf{S}(i+1)$, formed by $\mathbf{q}_o(i)$, $\mathbf{p}(i)$, and $\mathbf{q}_f(i)$ in joint space, find a minimum-time traveling path that the manipulator can follow within the prescribed deviation bound $e(i)$ at corner point $\mathbf{p}(i)$, subject to the limits on joint angular velocities, $|v_j| \le v^j_{\max}$, and bounds on joint torques/forces, $|u_j| \le u^j_{\max}$ $j = 1,\ldots,n$.

To obtain an analytical solution to the Problem LMTPP, one may attempt to apply Pontryagin's maximum principle with manipulator dynamics (to be defined as Eq.(3) later) and constraints on the state variables (i.e. on the angular velocities and position deviations at corner points) as well as on control inputs (i.e. bounds on joint torques/forces). Then, 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 manipulator in following segments $\mathbf{S}(i)$ $i = 1,\ldots,M$, 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 manipulators, it is almost impossible to obtain any analytic or numerical solution to the LMTPP Problem from the maximum principle. However, without appealing to direct use of the maximum principle, we can explore some intrinsic properties of this problem and find a suboptimal solution with an additional assumption.

A2. A constant acceleration, $\mathbf{a}_c(i)$, is assumed during each change_segment $\mathbf{CS}(i)$, $i = 1,2,\ldots,M$.

Assumption A2 is employed to simplify the analysis of the path deviations at corner points by utilizing the local acceleration bounds around corner points $\mathbf{p}(i)$ $i = 1,2,\ldots,M-1$. Note, however, that this assumption does not impose any unrealistic demand on the path-planning; if A2 is deemed unrealistic, one can divide $\mathbf{CS}(i)$ into a finite number of subregions within each of which a constant acceleration is then assumed.

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, consider the nonlinear, coupled manipulator dynamics. Using the Lagrangian formulation, the dynamics of the manipulator can be described by

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

where $\mathbf{u}$ is an $n\times 1$ generalized force/torque vector, $\mathbf{q}$, $\dot{\mathbf{q}}$, $\ddot{\mathbf{q}}$ are vectors of generalized coordinates, velocities, and accelerations, respectively. $\mathbf{D}(\mathbf{q})$ is an $n\times n$ inertial matrix, $\mathbf{h}(\mathbf{q},\dot{\mathbf{q}})$ is an $n\times 1$ viscous friction, Coriolis and centrifugal force vector, $\mathbf{g}(\mathbf{q})$ is an $n\times 1$ gravitational loading vector, and n is the number of joints of the manipulator. The inertia, the gravity loading, and the 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 manipulator's payload (i.e. tool and parts).

The constraints on torques/forces are related to acceleration by:

$$-\mathbf{u}_{\max} \le \mathbf{D}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{h}(\mathbf{q}, \dot{\mathbf{q}}) + \mathbf{g}(\mathbf{q}) \le \mathbf{u}_{\max} \tag{4}$$

where $\mathbf{u}_{\max} = (u^1_{\max}, u^2_{\max}, \ldots, u^n_{\max})^T$. If both $\mathbf{q}$ and $\dot{\mathbf{q}}$ are known, then bounds on $\ddot{\mathbf{q}}$ 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, [4] 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 $\mathbf{p}(i)$, $i = 1,2,\ldots,M-1$, we compute the parameters of manipulator dynamics for three distinct cases with velocities (i.e. $\dot{\mathbf{q}}$) $\mathbf{w}_{\max}(i)$, $\mathbf{w}_{\max}(i+1)$, and an average velocity $\mathbf{w}_{avg}(i)$, but with the same position, $\mathbf{q}_c(i) = \mathbf{p}(i) + \mathbf{e}(i)$,[5] where $\mathbf{e}(i)$ is defined as a path deviation vector around $\mathbf{p}(i)$ with magnitude $e(i)$, and with the same direction as the vector $-\mathbf{r}(i) + \mathbf{r}(i+1)$. $\mathbf{w}_{\max}(i)$ is the cruise velocity approaching $\mathbf{TR}(i)$, $\mathbf{w}_{\max}(i+1)$ is the cruise velocity departing from $\mathbf{TR}(i)$, and the average velocity, $\mathbf{w}_{avg}(i) = (\mathbf{w}_{\max}(i)+\mathbf{w}_{\max}(i+1))/2$, is used for approximating the velocity in the middle of change_segment $\mathbf{CS}(i)$. If it is desired to set a uniform acceleration bound around each corner point $\mathbf{p}(i)$, one can select bounds $c^j_{\max}(i)$ and $c^j_{\min}(i)$ during transition $\mathbf{TR}(i)$. That is,

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

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

Authorized licensed use limited to: University of Michigan Library. Downloaded on January 19,2023 at 21:13:41 UTC from IEEE Xplore. Restrictions apply.

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

where $\mathbf{a}(i)$ denotes the acceleration during $\mathbf{TR}(i)$, and

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

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

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

THE MINIMUM-TIME PATH PLANNING

The functional relationship of the proposed minimum-time path planner to others in the system is described in Fig. 2. With the preceding assumptions and discussions, the LMTPP(i) problem can be solved by the following steps for $i = 1,\ldots,M-1$ (see Fig. 3 for an illustration of these steps).[6]

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

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

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

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

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

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

$$\mathbf{r}_o(i) = \mathbf{r}(i); \qquad \mathbf{r}_f(i) = \mathbf{r}(i+1) \tag{7}$$

Then every vector can be uniquely decomposed into $\mathbf{r}_o(i)$ and $\mathbf{r}_f(i)$ components as follows (see Fig. 3).

$$\begin{aligned} \mathbf{e}(i) &= -e_o(i)\,\mathbf{r}_o(i) + e_f(i)\,\mathbf{r}_f(i)\\ \mathbf{a}_o(i) &= -a_o(i)\,\mathbf{r}_o(i)\\ \mathbf{a}_f(i) &= a_f(i)\,\mathbf{r}_f(i)\\ \mathbf{a}_c(i) &= -a_{co}(i)\,\mathbf{r}_o(i) + a_{cf}(i)\,\mathbf{r}_f(i) \end{aligned} \tag{8}$$

$$\begin{aligned} \mathbf{w}_o(i) &= w_o(i)\,\mathbf{r}_o(i)\\ \mathbf{w}_f(i) &= w_f(i)\,\mathbf{r}_f(i)\\ \mathbf{p}(i) - \mathbf{q}_b(i) &= \xi_o(i)\,\mathbf{r}_o(i)\\ \mathbf{q}_f(i) - \mathbf{p}(i) &= \xi_f(i)\,\mathbf{r}_f(i) \end{aligned}$$

where scalar quantities with subscripts o and f represent their $\mathbf{r}_o$ and $\mathbf{r}_f$ components, respectively. $\xi_o(i)$ and $\xi_f(i)$ denote the magnitudes of vectors $\mathbf{p}(i)-\mathbf{q}_b(i)$ and $\mathbf{q}_f(i)-\mathbf{p}(i)$, respectively. One can plot each transition component in state space as depicted in Fig. 4. In S4, the final condition $\mathbf{q}_f(i)$ is set as a position when the velocity reaches $\mathbf{w}_{\max}(i+1)$ with constant acceleration $\mathbf{a}_f(i)$. This value of $\mathbf{q}_f(i)$ is used as the initial position in the next segment, i.e., $\mathbf{q}_b(i+1) = \mathbf{q}_f(i)$. 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. 4, we can compute traveling time $T(i)$ and positions $\mathbf{q}_a(i)$, $\mathbf{q}_b(i)$, and $\mathbf{q}_d(i)$, as functions of $a_o(i)$, $a_{co}(i)$, $a_{cf}(i)$, $a_f(i)$, $w_o(i)$, and $w_f(i)$ (the dependence on segment i is omitted for notational simplicity in the sequel). For convenience, let

$$\tau = \sqrt{\frac{2e_o}{a_{co}}} + \sqrt{\frac{2e_f}{a_{cf}}} \qquad \text{and}$$

$$c_o = \frac{w_o}{2a_o} + \frac{w_f}{2a_f} + \frac{\xi_o}{w_o} + \frac{\xi_f}{w_f}.$$

i) When $a_{co}\tau < w_o$, $a_{cf}\tau < w_f$, i.e. Cruise - Deceleration - Change_segment - Acceleration - Cruise;

$$\begin{aligned} \mathbf{q}_a(i) &= \mathbf{p} - [\tfrac{1}{2}a_{co}(1 - \tfrac{a_{co}}{a_o})\tau^2 + \tfrac{w_o^2}{2a_o}]\mathbf{r}_o\\ \mathbf{q}_b(i) &= \mathbf{p} - \tfrac{1}{2}a_{co}\tau^2\mathbf{r}_o\\ \mathbf{q}_d(i) &= \mathbf{p} + \tfrac{1}{2}a_{cf}\tau^2\mathbf{r}_f\\ \mathbf{q}_f(i) &= \mathbf{p} + [\tfrac{1}{2}a_{cf}(1 - \tfrac{a_{cf}}{a_f})\tau^2 + \tfrac{w_f^2}{2_f}]\mathbf{r}_f\\ T(i) &= c_o + \tau(1 - \tfrac{a_{co}}{a_o} - \tfrac{a_{cf}}{a_f})\\ &\quad -\tau^2[\tfrac{a_{co}}{2w_o}(1 - \tfrac{a_{co}}{a_o}) + \tfrac{a_{cf}}{2w_f}(1 - \tfrac{a_{cf}}{a_f})] \end{aligned} \tag{9a}$$

ii) When $a_{co}\tau \ge w_o$, $a_{cf}\tau < w_f$, i.e. Cruise Change_segment - Acceleration - Cruise;

$$\begin{aligned} \mathbf{q}_b(i) = \mathbf{q}_a(i) &= \mathbf{p} - \frac{w_o^2}{2a_{co}}\mathbf{r}_o\\ \mathbf{q}_d(i) &= \mathbf{p} + \tfrac{1}{2}a_{cf}(\tfrac{w_o}{a_{co}})^2\mathbf{r}_f\\ \mathbf{q}_f(i) &= \mathbf{p} + [\tfrac{1}{2}a_{cf}(1 - \tfrac{a_{cf}}{a_f})(\tfrac{w_o}{a_{co}})^2 + \tfrac{w_f^2}{2a_f}]\mathbf{r}_f\\ T(i) &= c_o - \frac{w_o}{a_{co}}\frac{a_{cf}}{a_f} - (\frac{w_o}{a_{co}})^2(1 - \frac{a_{cf}}{a_f})\frac{a_{cf}}{2w_f} \end{aligned} \tag{9b}$$

iii) When $a_{co}\tau < w_o$, $a_{cf}\tau \ge w_f$, i.e. Cruise - Deceleration - Change_segment - Cruise;

$$\begin{aligned} \mathbf{q}_a(i) &= \mathbf{p} - [\tfrac{1}{2}a_{co}(1 - \tfrac{a_{co}}{a_o})(\tfrac{w_f}{a_{cf}})^2 + \tfrac{w_o^2}{2a_o}]\mathbf{r}_o\\ \mathbf{q}_b(i) &= \mathbf{p} - \tfrac{1}{2}a_{co}(\tfrac{w_f}{a_{cf}})^2\mathbf{r}_o\\ \mathbf{q}_d(i) = \mathbf{q}_f(i) &= \mathbf{p} - w_f^2/\,2a_{cf}\mathbf{r}_o \end{aligned}$$

[6]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.

Authorized licensed use limited to: University of Michigan Library. Downloaded on January 19,2023 at 21:13:41 UTC from IEEE Xplore. Restrictions apply.

$$T(i) = c_o - \frac{w_f}{a_{cf}} \frac{a_{co}}{a_o} - \left(\frac{w_f}{a_{cf}}\right)^2 \left(1 - \frac{a_{cf}}{a_o}\right) \frac{a_{cf}}{2w_o} \qquad (9c)$$

iv) When $a_{co}\tau \geq w_o$, $a_{cf}\tau \geq w_f$, i.e. Cruise Change_segment - Cruise;

$$\mathbf{q}_a(i) = \mathbf{q}_b(i) = \mathbf{p} - \frac{w_o^2}{2a_{co}}\mathbf{r}_o$$

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

$$T(i) = \frac{\xi_o}{w_o} + \frac{\xi_f}{w_f} \qquad (9d)$$

Now, consider the calculation of $\mathbf{a}_o(i)$ and $\mathbf{a}_f(i)$. We can easily show that $\frac{\partial T(i)}{\partial a_o(i)} < 0 \quad i = 1,\ldots,M-1$ for the cases i) and iii). This property implies that $a_o(i)$ 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 $\mathbf{a}_o(i)$ and $\mathbf{a}_f(i)$ so as to attain maximum magnitudes in the directions of $-\mathbf{r}_o(i)$ and $\mathbf{r}_f(i)$, respectively.

To determine $\mathbf{a}_c(i)$, which is a function of $a_{co}(i)$ and $a_{cf}(i)$, i.e., $\mathbf{a}_c(i) = a_{co}(i)\mathbf{r}_o(i) + a_{cf}(i)\mathbf{r}_f(i)$, we have to use the polyhedron boundary of the accelerations given by Eq. (5). Then, the problem becomes a two-dimensional nonlinear optimization minimizing Eq. (9) with respect to $a_{co}(i)$ and $a_{cf}(i)$ subject to linear constraints (5). For solution to this problem, we can get the following useful property.

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

It is easy to show $\frac{\partial T(i)}{\partial a_{co}(i)} \leq 0$ and $\frac{\partial T(i)}{\partial a_{cf}(i)} \leq 0$.

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 $\mathbf{a}_c(i)$ should be maximized. But the values of $a_{co}(i)$ and $a_{cf}(i)$ are interrelated with the linear constraints (5). Hence, the minimum of $T(i)$ occurs at the boundary of inequalities (5), resulting in a one-dimensional optimization problem.

Acceleration during change_segment $CS(i)$, $\mathbf{a}_c(i)$, can be obtained with a suitable bisection algorithm searching along the boundary of Eq. (5). Let ϑ be the angle between $\mathbf{a}_o(i)$ and $\mathbf{a}_f(i)$. The following iterations are performed for $k = 1,2,\ldots$. Choose a set of accelerations $\mathbf{a}_l$ having angles $[(2l-1)/2^k]\vartheta$, $l = 1,2,\ldots,2^{k-1}$, and the corresponding traveling times, T_kl's. Compare T_kl's and choose the minimum and set it to T_k. If the bisecting angle $2^{-k}\vartheta$ gets smaller and improvement in T_k from T_{k-1} becomes insignificant, then the algorithm terminates.

For (the first) segment S(0), we need a maximum acceleration until the velocity reaches $\mathbf{w}_{max}(1)$. No iteration is necessary in this segment. We can simply compute the maximum acceleration, and set $\mathbf{q}_f(0) = \mathbf{q}_b(1)$ for the next segment. For (the last) segment $S(M)$, we need cruise with $\mathbf{w}_{max}(M)$ followed by the maximum deceleration to the final position $\mathbf{p}(M)$ 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 as follows.

1. Set $i = 0$.
    1.1. Compute the maximum acceleration $\mathbf{a}_f(0)$.
    1.2. Compute traveling time $T(0)$
    1.3. Compute the initial position for $S(1)$, $\mathbf{q}_b(1)$.

2. Set $i = i + 1$, then
    2.1. Compute unit vectors $\mathbf{r}_o(i)$ and $\mathbf{r}_f(i)$.
    2.2. Compute a position, $\mathbf{q}_c(i)$, with the maximum path deviation during transition.
    2.3. Compute velocity bounds for $\mathbf{w}_o(i)$ and $\mathbf{w}_f(i)$.
    2.4. Compute maximum deceleration $\mathbf{a}_o(i)$ along $S(i)$, and maximum acceleration $\mathbf{a}_f(i)$ along $S(i+1)$.
    2.5. Compute acceleration $\mathbf{a}_c(i)$ during change_segment $CS(i)$, and traveling time $T(i)$ by using the bisecting algorithm.
    2.6. Compute the initial position for the next segment $\mathbf{q}_b(i+1)$.
    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 $\mathbf{a}_o(M)$ toward $(\mathbf{p}(M),0)$
    3.2. Compute the traveling time $T(M)$.

4. Compute the total traveling time $T_{total} = \sum_{i=0}^{M} T(i)$

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

A PATH PLANNING EXAMPLE

Using a simulator of the PUMA 600 manipulator on DEC VAX 11/780, we have conducted a simulation of 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 Eq.(10) with the coordinate system using the Denavit and Hartenberg representation in Table 1, 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.) For $i=1,2,\cdots,n$

$$\sum_{j=1}^{n} d_{ij}(\mathbf{q})q_j + \sum_{j=1}^{n}\sum_{k=1}^{n} h_{ijk}(\mathbf{q})q_j q_k + g_i(\mathbf{q}) = u_i \qquad (10)$$

Typical terms have the following form:

$$\begin{aligned} d_{11}(\mathbf{q}) = {} & J_{111} + J_{133} + J_{233} + 2J_{234}d_2 + J_{244}d_2^2 \\ & + J_{322} + J_{333} + J_{334}d_2^2 \\ & + (J_{211} + 2J_{214}a_2 + J_{244}a_2^2 + J_{344}a_2^2)\,C_2^2 + J_{222}S_2^2 \\ & + (J_{311} - J_{333})\,C_{23}^2 + 2J_{334}a_2C_2S_{23} \end{aligned}$$

$$\begin{aligned} h_{211} = {} & (J_{211} - J_{222} + 2J_{214}a_2 + J_{244}S_2 + J_{344}a_2^2)\,S_2C_2 \\ & + (J_{311} - J_{333})\,C_{23}S_{23} - J_{334}a_2\,(C_2C_{23} - S_2S_{23}) \end{aligned}$$

$$g_2 = -(a_2m_3 + a_2m_2 + m_2a_{\overline{co}2})\,C_2g - m_3\bar{z}_3S_{23}g$$

where J_{ijk} is the $(j,k)-th$ element of the 4x4 inertia tensor, J_i, for the $i-th$ joint; d_2 a_2 are lengths related to the arm coordinate frame; and for $i, j = 1,2,3$
$C_i = cos(q_i)$, $S_i = \sin(q_i)$ $C_{ij} = \cos(q_i+q_j)$
$S_{ij} = \sin(q_i+q_j)$

Authorized licensed use limited to: University of Michigan Library. Downloaded on January 19,2023 at 21:13:41 UTC from IEEE Xplore. Restrictions apply.

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 (see Table 2). These are approximate figures acquired from the manufacturer's specification.

As shown in Table 3, for simulation we selected a set of corner points $\mathbf{p}(i)$, $i = 0,1,\ldots,M$, and path deviation bounds at these corner points $e(i)$, $i = 1,\ldots,M-1$. The bounds on the control input torques are assumed to be: $|u_1| \le 100\,Nm$, $|u_2| \le 150\,Nm$, and $|u_3| \le 50\,Nm$, and the maximum angular velocity is set to $90° / second$ 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. $\mathbf{q}$ and $\dot{\mathbf{q}}$) 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. The corresponding simulation results are given in Table 4. 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 is simulated and presented in Table 5. Here, all $e(i)$'s are set to an equal value for $i = 1,\ldots,M-1$. 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 $\mathbf{e}(i)$, $i=1,2,\ldots,M$.

As a whole, for the example considered here our minimum-time path planning has indeed shown a significant improvement in the total traveling time. Consequently, the path planning method has high potential for enabling the controller to efficiently operate the manipulator to its maximum capabilities.

CONCLUDING REMARKS

We have developed a minimum-time path planning method in joint space with manipulator dynamics included. An absolute path deviation bound for each corner point can be specified as a design variable. Local upper bounds on joint accelerations are derived from manipulator dynamics so as to nearly fully utilize 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 (i) a large number of variables should be considered simultaneously, and (ii) 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 sub-regions 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 manipulator 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).

Path planning has not received much attention despite its importance. Particularly, there are numerous publications in the area of manipulator control or path control or path 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 manipulator 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*, third edition, McGraw-Hill, Inc., 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", *Proc. 22nd Conference on Decision and Control*, San Antonio, Tx., pp. 1199-1204, Dec. 1983. (Also will appear in *IEEE Trans. Automatic Control.*)

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

[6] J. Y. S. Luh, M. W. Walker, and R. P. C. Paul, "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.

Authorized licensed use limited to: University of Michigan Library. Downloaded on January 19,2023 at 21:13:41 UTC from IEEE Xplore. Restrictions apply.

[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", *Proc. Joint Auto., Contr., Conf.*, vol. 1, pp. 124-131, 1977.

[10] C. S. Lin, P. R. Chang, and J. Y. S. Luh, "Formulation and optimization of cubic polynomial joint trajectories for mechanical manipulators", *IEEE Conf. on Decision and Control*, Orlando, Florida, Dec. 1982. Also appeared in *IEEE Trans. Automatic Control*, 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. Automatic Control*, vol. AC-28, No. 2, pp. 133-153, February 1983.

[13] T. Lozano-Perez, "Spatial planning: A configuration space approach", *IEEE Trans. on Computers*, 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", *Proc. 22nd Conference on Decision and Control*, San Antonio, TX, pp. 1449-1457, Dec. 1983.

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 1. Link coordinate system for a PUMA 600 robot.

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 2. Mass, first moments, and inertias of the first three joints for the PUMA 600 manipulator.

i	Corner points, $p(i)$			Tolerances
	Joint 1 (deg.)	Joint 2 (deg.)	Joint 3 (deg.)	$\varepsilon(i)$ (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

Table 3. Corner points and tolerances in path deviation used for simulation.

	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 4. Traveling time.

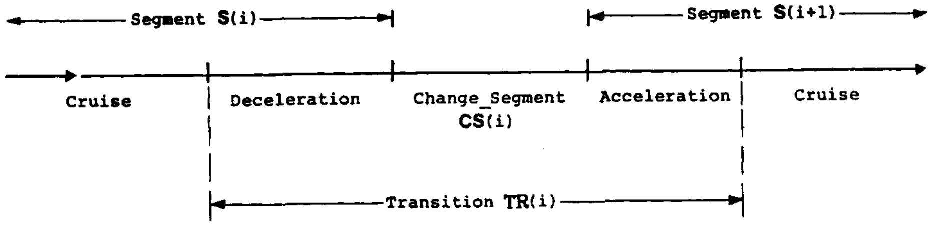


Figure 1. Transition TR(i) and Change_Segment CS(i).

Authorized licensed use limited to: University of Michigan Library. Downloaded on January 19,2023 at 21:13:41 UTC from IEEE Xplore. Restrictions apply.

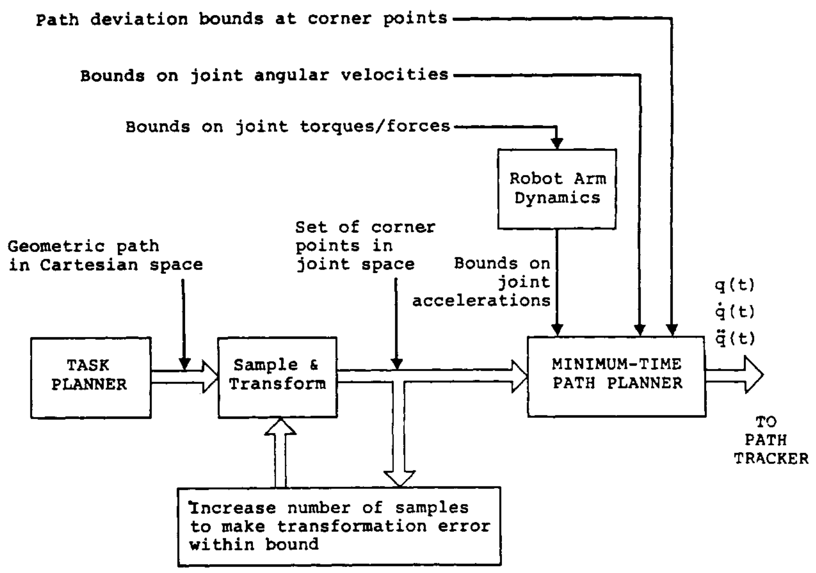

Path deviation bounds at corner points
Bounds on joint angular velocities
Bounds on joint torques/forces
Robot Arm Dynamics
Geometric path in Cartesian space
Set of corner points in joint space
Bounds on joint accelerations
TASK PLANNER
Sample & Transform
MINIMUM-TIME PATH PLANNER
q(t)
q̇(t)
q̈(t)
TO PATH TRACKER
Increase number of samples to make transformation error within bound

Figure 2. Description of input and output of the minimum-time path planner

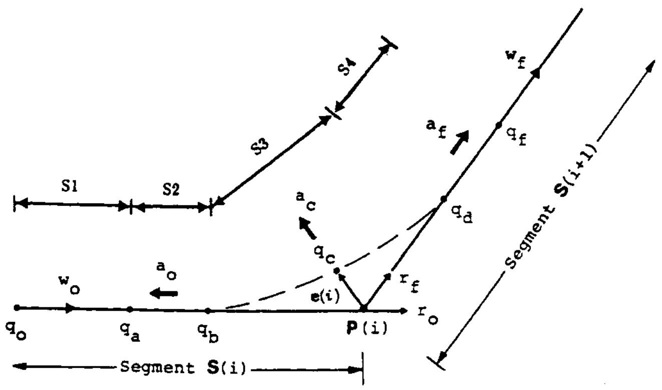
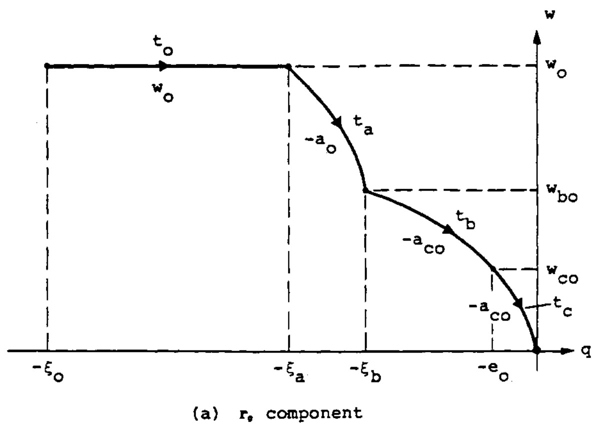
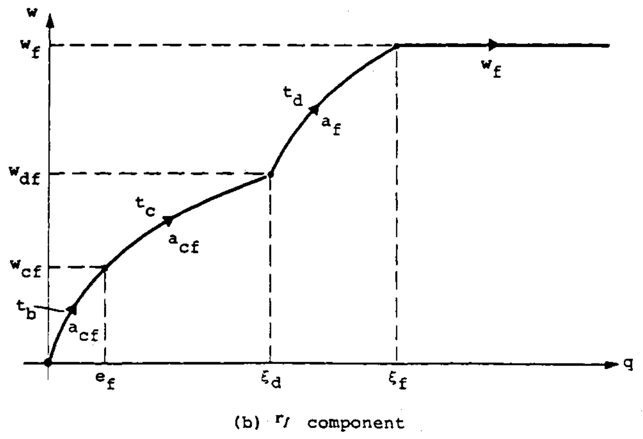


Figure 3 Individual steps in transition TR(i).



(a) r_o component



(b) r_f component

Figure 4. Transition components in state space.

Authorized licensed use limited to: University of Michigan Library. Downloaded on January 19,2023 at 21:13:41 UTC from IEEE Xplore. Restrictions apply.