MASSACHUSETTS INSTITUTE OF TECHNOLOGY

ARTIFICIAL INTELLIGENCE LABORATORY

A.I. Memo No. 804 November 1984

Planning of Minimum-Time Trajectories for Robot Arms

Gideon Sahar and John M. Hollerbach

Abstract: The minimum-time path for a robot arm has been a long-standing and unsolved problem of considerable interest. We present a general solution to this problem that involves joint-space tesselation, a dynamic time-scaling algorithm, and graph search. The solution incorporates full dynamics of movement and actuator constraints, and can be easily extended for joint limits and workspace obstacles, but is subject to the particular tesselation scheme used. The results presented show that, in general, the optimal paths are not straight lines, but rather curves in joint-space that utilize the dynamics of the arm and gravity to help in moving the arm faster to its destination. Implementation difficulties due to the tesselation and to combinatorial proliferation of paths are discussed.

Acknowledgments: This paper describes research done at the Artificial Intelligence Laboratory of the Massachusetts Institute of Technology. Support for the laboratory's research is provided in part by the Systems Development Foundation, in part by the Office of Naval Research under contract N00014-81-0494, and in part by the Defense Advanced Reserach Projects Agency under Office of Naval Reserach contracts N00014-80-C-0505 and N00014-82-K-0334.

1 Introduction

The determination of the time-optimal path for manipulators is an important problem in robot trajectory planning. This paper presents a general solution to this problem involving *joint space* tesselation and graph search that is made relatively efficient through use of time-scaling properties of dynamics. The minimum-time requirement imposes constraints on the velocities at each point on our grid, so that tesselation in velocity space is not needed. Our algorithm takes into account a full dynamic model of a manipulator and actuator torque limits in arriving at the time-optimal trajectory, i.e. the path and the time dependence along the path, from any given starting state to any final state. Kinematic constraints due to joint limits and the presence of obstacles are quite easily incorporated into the solution.

Attempts in the past at solving this problem have fallen short of a general solution. Most commonly, researchers have linearized the dynamics in order to apply standard techniques of linear optimal control theory to the solution. The earliest attempt along these lines appeared in (Kahn, 1969; Kahn and Roth, 1971), where the expected bang-bang solution with multiple switching points was derived. Approaches based on dynamics linearization have recently been cast into doubt due to time-scaling properties of dynamics (Hollerbach, 1983a, 1983b, 1984), since it can be shown that the velocity product terms have the same significance relative to the acceleration dynamic terms for all speeds of movement. Thus the main presumption used to justify linearization, namely that the velocity product terms can be ignored because they are only significant at higher movement speeds, is fundamentally wrong.

A second class of solutions do take into account the full dynamics but presume a bang-coast-bang form of control, again by analogy to the linear optimal case. Scheinman and Roth (1984) assume a bang-coast-bang control form for each joint where (1) each control has a maximum of two switching points and (2) the total positive torque time is equal to the total negative torque time. Their solution is an iterative one, where they set a time, solve for the distances covered and the velocities attained, and adjust the switching times until a satisfactory solution is found. Our results indi-

cate that these presumptions about control are in general not valid for the true time-optimal solution.

Brown (1984) independently developed an approach based on a *state space* tesselation and graph search, which nevertheless differs from our approach in critical ways. Brown presumes a bang-coast-bang solution with fixed switching points; all possible combinations of bangs and coasts yield 9 distinct torque patterns. At a given point in state space all 9 torque patterns are applied for a set period of time, and the dynamics are integrated to find the resultant state space point. The resultant state space point is rounded to the nearest neighbour in the tesselation. The network is searched via standard procedures to find the fastest trajectory. Again, the presumption on control makes his solutions non-optimal.

Purely kinematic approaches were used by Luh and Walker (1977), Luh and Lin (1981), and Lin, Chang and Luh (1983), who attempt to find the sequence of time intervals that minimize the total time spent on moving between two points in space. The three approaches differ in the details of the algorithms, but they all suffer from the same shortcomings: the dynamics of the arm are not considered at all, so that the constraints consist of sets of bounds on position, velocity, and acceleration. These bounds are imposed by the weakest configuration of the arm, so that the motion in other areas of the workspace is sub-optimal. In addition, all three algorithms require as input a set of knot points, which define a path. This path is not necessarily the best possible one.

The new development that facilitates the approach presented in this paper is the discovery of a fundamental time-scaling property of manipulator dynamics. This property was utilized to solve for the time-optimal trajectory along a predetermined path by (Bobrow, 1982; Bobrow, Dubowsky, and Gibson, 1983; Dubowsky and Shiller, 1984). The equations of motion are written in terms of the distance along the Cartesian path taken by the arm, and the bounds on the torques/forces obtainable from the motors are used to construct a curve in distance-velocity space that constitutes an upper limit on the performance of the arm. A set of switching points is then found that move the arm as close as possible to the limit curve. In his Ph.D. thesis, Bobrow proves the optimality of this algorithm. One actuator is always saturated, and the others adjust their torques so that

some constraints on the motion are not violated. The only problem with it is that the path has to be known beforehand. A very similar approach was developed later by Shin and McKay (1983), with the only difference being a parametrization of a path in joint space instead of in Cartesian space.

Hollerbach (1983a, 1983b, 1984) independently developed the time-scaling property of dynamics in joint space and used a simplified form of this property to scale a fixed velocity profile for maximum speed given actuator constraints. This general formulation has been discretized for our optimization algorithm.

The next section introduces the problem in a more rigorous fashion, and describes the way with which we manage to work in *state space*, yet tesselate only in *joint space*. Section 3 introduces the Dynamic Scaling algorithm, which is used to calculate the times of travel along each segment of the grid, along with the required torques and the possible velocities at the end of the segment, given initial conditions on position and velocity at the beginning of the segment. Section 4 presents results of a 2-D simulation, and section 5 concludes with a discussion.

2 The General Solution

The objective is to move the arm from point A to point B in state space in minimum time. Given are:

- the kinematic and inertial parameters of the manipulator,
- the dynamic equations

$$\tau = \mathbf{H}(\theta(t))\ddot{\theta}(t) + \dot{\theta}(t)^T\mathbf{C}(\theta(t))\dot{\theta}(t) + \mathbf{g}(\theta(t)), \qquad (1)$$

  where τ is the vector of joint torques, θ is the n-dimensional joint-space position vector, $\mathbf{H}$ is a $n \times n$ symmetric inertia matrix, $\mathbf{C}$ is a $n \times n \times n$ tensor, n is the number of joints, and $\mathbf{g}$ is a gravity dependent vector, and

- constant bounds on the torques/forces available from the motors,

$$\tau_l \leq \tau \leq \tau_u, \qquad (2)$$

which can be made state dependent without modifying the algorithm.

The main idea behind our algorithm is to tesselate *joint space* into a grid. The requirement for time-optimality, together with the dynamics of the arm, constrain the number of velocities possible at each position node, given the position and velocity at the previous node, so that there is no need to tesselate velocity space. A tree of all possible state-space paths can be constructed, and then searched for the minimum-time path.

In terms of assigning velocity tesselations to each position node, the results of (Bobrow 1982) show that one actuator is always at a torque bound. Therefore, there are at most $2n$ possible transitions from a given state to a neighbouring position. The time scaling algorithm presented in the next section determines for each transition the velocity at this neighbouring position and the transition time. Many of these $2n$ transitions can be discarded immediately, due either to saturation in one of the actuators causing the torque in one of the other motors to exceed its bound, or to a contradictory situation where the velocity is in the direction opposite to the position increment. In our 2-D implementation, the number of solutions at each state space point was usually two or less.

2.1 Graph Search

The graph search can be made considerably more efficient through the use of some general search techniques: (1) best first search, which chooses to expand the best path so far, and (2) branch and bound, which uses an upper bound on the travel time to eliminate any path with a partial cost larger than the upper bound. The choice here was to first run the algorithm with a very sparse grid, and use the time found, rounded up in some suitable manner which depends on the magnitude of the result, as the upper bound for a second run with a finer grid.

The procedure for finding the optimal path follows.

1. Form a queue of paths, containing an empty path (one with only the starting node).

2. Until goal is reached (success), or the queue is empty (failure):

(a) Remove the first path from the queue.

(b) Calculate the positions of the next points reachable from the current (last in path) point.

(c) For each one of the next points, the dynamic scaling algorithm, as will be described in the next section, is used to find the possible velocities at the next points, and the times of travel.

(d) New paths are formed from the old path and from all admissible new points (a point is inadmissible if one of the calculated torques exceeds its bound, if the velocity of one joint is in the opposite direction to its position increment, or the current total time of travel exceeds the upper bound.)

(e) Add new paths to the queue and sort it such that the path with the smallest cost is on top.

3 The Dynamic Scaling Algorithm

The previous section described the global solution to finding the optimal path. This section describes the local solution for the minimum traveling time between nodes, the achieved velocity for each permissible transition, and the corresponding torques. We rely on a reformulation of the dynamics of a manipulator, which indicates how the underlying dynamics change when the time dimension of a trajectory changes (Hollerbach, 1984). Suppose the time dependence along a fixed path is changed from $\boldsymbol{\theta}(t)$ to $\boldsymbol{\theta}(r(t))$, where the time scaling factor $r(t)$ is a strictly increasing function of time with $r(0) = 0$ and $r(t_1) = t_f$ for some $t_1 > 0$. Then the joint torques $\boldsymbol{\tau}'(t)$ for the time-scaled movement are related to the original movement torques $\boldsymbol{\tau}(t)$ by:

$$\boldsymbol{\tau}'(t) = \dot{r}^2[\boldsymbol{\tau}(r) - \mathbf{g}(\boldsymbol{\theta}(r))] + \ddot{r}\,\mathbf{H}(\boldsymbol{\theta}(r)) \cdot \frac{d\boldsymbol{\theta}(r)}{dr} + \mathbf{g}(\boldsymbol{\theta}(t)) \qquad (3)$$

This general formulation is now discretized, with known position $\boldsymbol{\theta}(i-1)$ and $\boldsymbol{\theta}(i)$, respectively, at the beginning and end of the segment, and known velocity $\dot{\boldsymbol{\theta}}(i-1)$ at the beginning of the segment. Instead of looking at the

beginning of the segment (i.e. point $i-1$), consider the mid-point. Define an average velocity and acceleration

$$\dot{\boldsymbol{\theta}}(i-\frac{1}{2}) = \frac{\Delta\boldsymbol{\theta}(i)}{\Delta t}, \tag{4}$$

$$\ddot{\boldsymbol{\theta}}(i-\frac{1}{2}) = \frac{\dot{\boldsymbol{\theta}}(i)-\dot{\boldsymbol{\theta}}(i-1)}{\Delta t}. \tag{5}$$

where the distance $\Delta\boldsymbol{\theta}(i) = \boldsymbol{\theta}(i) - \boldsymbol{\theta}(i-1)$ is covered in time Δt. The important point here is that no functional relationship is assumed between $\dot{\boldsymbol{\theta}}(i-\frac{1}{2})$ and $\ddot{\boldsymbol{\theta}}(i-\frac{1}{2})$, but rather that these relationships are used as predictors that could be subject to correction. By assuming a linear velocity (constant acceleration) we can find the velocity at the end of the segment:

$$\dot{\boldsymbol{\theta}}(i) = 2\frac{\Delta\boldsymbol{\theta}(i)}{\Delta t} - \dot{\boldsymbol{\theta}}(i-1) \tag{6}$$

Substituting this into our definition of the average acceleration, we obtain:

$$\ddot{\boldsymbol{\theta}}(i-\frac{1}{2}) = 2\frac{\Delta\boldsymbol{\theta}(i)}{\Delta t^2} - 2\frac{\dot{\boldsymbol{\theta}}(i-1)}{\Delta t} \tag{7}$$

Suppose the time of travel was Δs instead of Δt. The new average velocity and acceleration at the mid-point are related to the old by

$$\dot{\boldsymbol{\theta}}'(i-\frac{1}{2}) = \frac{\Delta t}{\Delta s}\dot{\boldsymbol{\theta}}(i-\frac{1}{2}) \tag{8}$$

$$\ddot{\boldsymbol{\theta}}'(i-\frac{1}{2}) = \frac{\Delta t^2}{\Delta s^2}\ddot{\boldsymbol{\theta}}(i-\frac{1}{2}) + 2\frac{\Delta t-\Delta s}{\Delta s^2}\dot{\boldsymbol{\theta}}(i-1). \tag{9}$$

We call these quantities "scaled". If the scaled velocity and acceleration are substituted into the equations of motion (1), an equation for $\boldsymbol{\tau}'$, the torque required to cover the distance in Δs, in terms of $\boldsymbol{\tau}$, the torque that was used when the distance was covered in Δt, is obtained:

$$\boldsymbol{\tau}' = \frac{\Delta t^2}{\Delta s^2}(\boldsymbol{\tau}-\mathbf{g}) + 2\frac{\Delta t-\Delta s}{\Delta s^2}\,\mathbf{H}\cdot\dot{\boldsymbol{\theta}}(i-1) + \mathbf{g} \tag{10}$$

Rewriting equation (10) as a quadratic equation for Δs:

$$\Delta s^2(\boldsymbol{\tau}'-\mathbf{g}) + 2\Delta s\mathbf{H}\cdot\dot{\boldsymbol{\theta}}(i-1) - [\Delta t^2(\boldsymbol{\tau}-\mathbf{g}) + 2\Delta t\mathbf{H}\cdot\dot{\boldsymbol{\theta}}(i-1)] = 0 \tag{11}$$

These are n equations for the unknown Δs, one for each joint, which are found by substituting the torque bounds for τ'. They all have to be satisfied simultaneously in order to achieve optimality without violating one of the constraints. This development leads to a simple procedure to calculate the minimal time to travel from point $i-1$ to point i, and the combination of the torques that will do it:

1. Calculate $\theta(i-1/2)$, $\dot{\theta}(i-1/2)$, and $\ddot{\theta}(i-1/2)$, using the known $\theta(i-1)$, $\Delta\theta(i)$, and $\dot{\theta}(i-1)$, and an arbitrarily chosen Δt (setting $\Delta t = 1$ is the most efficient).

2. Solve the equations of motion (1) for the corresponding τ.

3. For each joint, use equation (11) to find Δs, where τ' is the vector of given torque bounds. Do it for every one of the bounds.

4. Use all the Δs found in step 3, together with the corresponding torque bound, to calculate the torques for the other joints from equation (10). Retain only those time/torque combinations that do not exceed the bounds.

5. The permited velocities at the end of the segment can be calculated from equation (6), with Δt replaced by Δs.

4 Implementation

The minimum-time algorithm is illustrated by simulation with a two degree of freedom, planar manipulator (Horn, 1975; Brady, Hollerbach, Johnson, Lozano-Perez, and Mason, 1982, chapter 1) on a Symbolics 3600 Lisp Machine. The equations of motion are

$$\begin{aligned}\tau_1 = {} & [I_1 + I_2 + \frac{1}{4}(m_1 l_1^2 + m_2 l_2^2) + m_2 l_1^2 + m_2 l_1 l_2 c_2]\ddot{\theta}_1 \\ & + (I_2 + \frac{1}{4}m_2 l_2^2 + \frac{1}{2}m_2 l_1 l_2 c_2)\ddot{\theta}_2 - \frac{1}{2}m_2 l_1 l_2 s_2 \dot{\theta}_2^2 - m_2 l_1 l_2 s_2 \dot{\theta}_1 \dot{\theta}_2 \\ & + [\frac{1}{2}m_2 l_2 c_{1+2} + l_1(\frac{1}{2}m_1 + m_2)c_1]g\end{aligned} \tag{12}$$

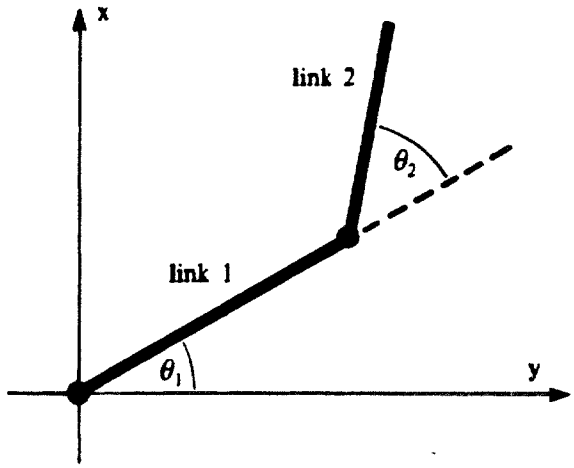


Figure 1: A 2 degree of freedom planar arm. $l_1 = l_2 = 0.5m$, $m_1 = 50kg$, $m_2 = 30kg$, $I_1 = 5kg \cdot m^2$, $I_2 = 3kg \cdot m^2$.

$$\tau_2 = (I_2 + \frac{1}{4}m_2 l_2^2 + \frac{1}{2}m_2 l_1 l_2 c_2)\ddot{\theta}_1 + (I_2 + \frac{1}{4}m_2 l_2^2)\ddot{\theta}_2 + \frac{1}{2}m_2 l_1 l_2 s_2 \dot{\theta}_1^2 + \frac{1}{2}m_2 l_2 c_{1+2}\, g \tag{13}$$

where θ_i, l_i, m_i, and I_i are the joint angle, length, mass, and moment of inertia, respectively, of link i, $s_i(c_i)$ are the sine(cosine) of joint i variable, and c_{1+2} is $\cos(\theta_1 + \theta_2)$. The values for the link lengths, masses, and moments of inertia were taken from (Asada, 1984) (Fig. 1). The torque bounds were chosen as

$$\begin{aligned} -350Nm \leq \tau_{\theta_1} \leq 350Nm \\ -100Nm \leq \tau_{\theta_2} \leq 100Nm\ . \end{aligned} \tag{14}$$

4.1 Tesselation of joint space

The manner of tesselation is illustrated in Fig. 2 for this two-link manipulator. The joint positions are first tesselated by connecting the beginning and final nodes by a straight line and dividing this line into k equal intervals. The remainder of the position grid is defined normal to this line by translating the tesselated points at integer multiples of the point spacing. From a given position node three paths are allowed: one path is parallel to

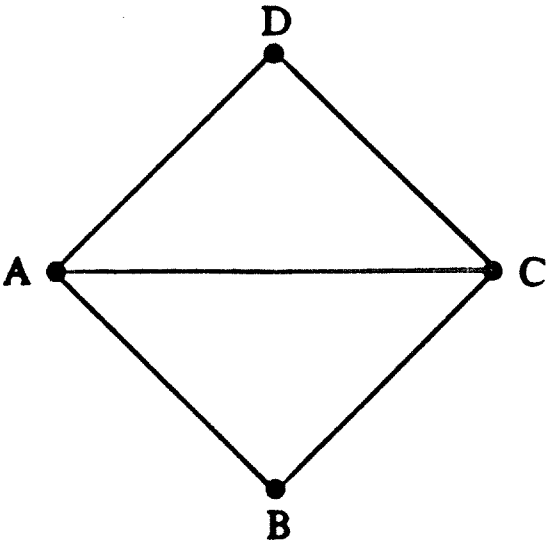


Figure 2: The basic grid unit

the original straight line and skips one grid point (from A to C), the other two paths connect diagonally relative to the first path in the direction of the goal (from A to B and A to D). See Fig. 3 for an example of a 3×3 grid.

The number of paths leaving each node is limited to 3 as a compromise between representation of path curvature and combinatorial proliferation of possible paths. This did create difficulties due to discontinuities, which were handled with special measures described later. Given the influence of the choice of the tesselation number k on the number of paths and hence on computation time, a usual value of $k = 15$ was used. The largest value of k used was 20. To further reduce the search, the two corners of the rectangular grid not on the diagonal connecting the start and the goal were cut away as it was noticed that the optimal paths generated by the planner tended to concentrate around the diagonal.

4.2 The problem of corners

One implication of tesselating space into a rectangular grid is that the motion may be discontinuous: when moving from point A to C through B (Fig. 2), the velocity has to drop to zero at point B, or change direction instantaneously, which implies infinite accelerations. Therefore the corner has to be smoothed out by, for example, fitting a circle between points A and

C. Since our local algorithm is defined for a straight line, the circle would have to be approximated by a sequence of straight lines. This solution is impractical because of the combinatorial expansion of states along the straight line sequence from start to goal, given the possible new states with each application of the time scaling algorithm.

Without corner smoothing, the velocity vector at the corner point B (Fig. 2), for example, is not aligned with the direction vector A to C. Globally, the arm has to decelerate sharply after the corner B to reach point C, and the effect on the algorithm is to discard any non-straight-line path as too costly. This technical problem with our tesselation is unacceptable since it is unlikely that a joint-space straight-line path is always optimal.

A heuristic approximation was used of redefining the direction but not the magnitude of the velocity vector at the corner point B to lie along B-C. For the rationale, assume that the true path between A and C is indeed a circular arc and that the velocity along the arc is approximately constant. Now let the radius of the circle shrink gradually, until the circle becomes a point at B. The accelerations required to keep the arm on the circle grow, until in the limit of the point they become infinite. The magnitude of the velocity, however, does not change, so that the magnitude of the velocity going into the circular arc is the same as the velocity magnitude coming out of the other side of the arc. In the limit, the same is true of point B.

4.3 Complexity

By counting the possible number of paths on a grid, such as the one in Fig. 3, the worst-case complexity for our planar example is estimated as $O(4^k)$, where k is the number of points along one dimension of the grid. This calculation assumes that only two velocities are possible at the end of each segment, given some initial conditions at the beginning of the segment. This assumption is based on observations during the simulation of many examples. It is a conservative assumption, inasmuch as in many segments only one possible velocity was retained, whereas there were never more than two.

Another complexity factor is the sorting time for the least costly path in the queue. With queues of thousands of paths the sorting operation re-

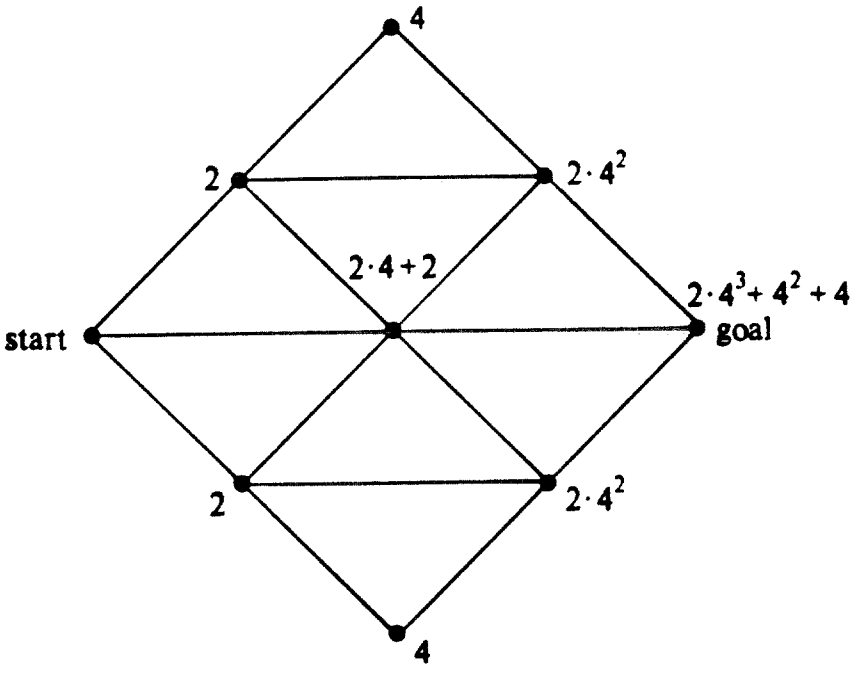


Figure 3: Number of possible trajectories at each node

quires minutes, which makes this algorithm too impractical. Based on this experience, the sorting operation was eliminated and the new path was inserted in the appropriate place in the already sorted queue. In our Lisp Machine implementation the improvement by simple merging was somewhat offset by creation of extraneous lists, leading to address space and garbage collection problems that increased running overhead by a half.

4.4 Results

The results of two simulated runs of the algorithm are shown, with initial and final velocities taken to be zero. Figures 4–6 show the path for moving from $[-0.5, -1.0]$ to $[0.5, 1.0]$ in joint space, which took 0.836 seconds of simulated time. The figures illustrate the tendency of the algorithm to find a path which is not a straight line, neither in joint-space nor in Cartesian space, but is not too far from being one in joint space. The best path tends to be one where the arm rearranges slightly its configuration, so that dynamically it will be easier to move. Therefore, joint θ_1 starts by moving backwards. Figure 6 shows the magnitudes of the acceleration and velocity

in Cartesian space. Note the similarity to a bang-coast-bang profile, in terms of acceleration.

In contrast, moving from a fully extended position ($[0.0, 0.0]$ in joint space) along the Cartesian x-axis, to the point $[x, y] = [0.5, 0.0]$ ($[-\pi/3, 2\pi/3]$ in joint space), results in a straight line, both in Cartesian space and in joint space (Figures 7–9). This is the result of the particular configuration of the arm, where the elbow is pointing down, and gravity does not have much effect on the motion. Therefore, the fastest path is also the shortest one, a straight line in joint space, which for this configuration happens also to be a straight line in Cartesian space. This motion was accomplished in 0.525 seconds of simulated time.

From these, and many more results emerges a pattern: usually, the fastest path is close to a straight line in joint space. Although the grid was not fine enough to be able to say so conclusively, it seems that the path is symmetric about the mid-point between the starting point and the destination. The torques tend to switch twice at the most, so one heuristic used by Scheinman and Roth (1984) seems to be justified (see the Introduction), but contrary to their other heuristics, only one joint torque is at a maximum at a time, and the total positive torque time is not equal to the total negative torque time.

5 Discussion and Conclusions

A method for finding minimum-time paths for manipulator arms, given dynamic, kinematic, and geometric constraints was developed. The method is based on the creation of a state-space search tree representing all possible solutions, and searching it for the best one. The search tree is smaller than one would initially expect, because velocity space was not actually tesselated. Instead, it was recognized that the requirement for time-optimality limited the number of possible velocities at the end of each segment.

The algorithm as currently formulated and implemented is not suitable for routine off-line trajectory planning, as even a 15×15 grid required minutes to hours of computation. We see the current primary benefit as providing the first idea of what the minimal time path looks like, however long the computation time, which was not possible by any means before.

From this knowledge it would be possible to compare the performance of more efficient trajectory planning methods and perhaps develop a method that is faster but sufficiently close to optimal. It might be possible to improve the algorithm itself by new graph search methods or hierarchical methods that zero in on the optimal path while restricting the state space expansion. The Lisp Machine itself is quite slow for numerical operations, and a more algebraically tuned system might speed the algorithm considerably. Finally, it may be possible to recast the algorithm into a faster parallel computation.

Several methods were examined for pruning the search tree of possible paths, but further improvements are required. Ideally, one would like a lower bound estimate of the cost from any point in state space to the goal in order to aid graph pruning (Winston, 1984), by a process of comparing the sum of this estimate and the current cost at the state space node to the upper bound. If a path can be thrown out early, then no time is wasted on its descendents. We had considered as a lower bound the time to travel from the present node to the goal while accelerating only and without endpoint velocity constraint, but we cannot prove this bound is valid. It seems that a true lower bound would presume some form of analytic solution to the very problem we are trying to solve.

The present tesselation, which allows from each node only three directions of motion, may exclude better paths. It would be desirable to find better tesselation techniques that allow finer resolution of direction and distance, yielding better approximation of curves by straight line segments, yet curbing the combinatorial proliferation of nodes and retaining efficient management of the grid. With a finer direction tesselation, the corner problem would be diminished. Lacking that, better solutions to the corner problem are required, since the present approximation implies unrealizable infinite accelerations at the corner point. Along the same lines, the current restriction to directions where at least one joint moves closer to the goal should be relaxed, since conceivably a solution might involve temporarily moving diametrically away from the goal.

The algorithm is easily extended to handle configuration constraints due to joint excursion limits and to workspace obstacles, when the latter are converted to obstacles in joint space (e.g., Lozano-Perez, 1982). Since these

translate into forbidden regions in joint space, the grid nodes within the forbidden regions are excluded from the tesselation. The ease of incorporating obstacle avoidance is a general feature of joint space tesselation, as also noted by (Brown, 1984), and is an advantage over trajectory representation by parameterized paths.

Finally, the algorithm presented here indicates that the fastest motion between two points is close to a straight line in joint space. The curvature at the beginning and end of the path in Fig. 4 may represent an interaction between the shortest path in joint space and the natural dynamics of the movement. Presently, it cannot be completely ruled out that tesselation problems contributed to some extent in forcing a nearly straight-line solution in joint space.

The complexity of the state-space search may currently rule out six degree of freedom implementations, but an implementation on a three joint manipulator seems possible. Conceivably a sparser grid might yield knot points for conventional polynomial interpolation that would nevertheless give superior paths. In addition, our algorithm can serve as a starting point for the development of faster, less exact algorithms, and for the verification of their results.

6 Acknowledgments

This paper describes research done at the Artificial Intelligence Laboratory of the Massachusetts Institute of Technology. Support for the laboratory's research is provided in part by the Systems Development Foundation, in part by the Office of Naval Research under contract N00014-81-0494, and in part by the Defense Advanced Reserach Projects Agency under Office of Naval Reserach contracts N00014-80-C-0505 and N00014-82-K-0334.

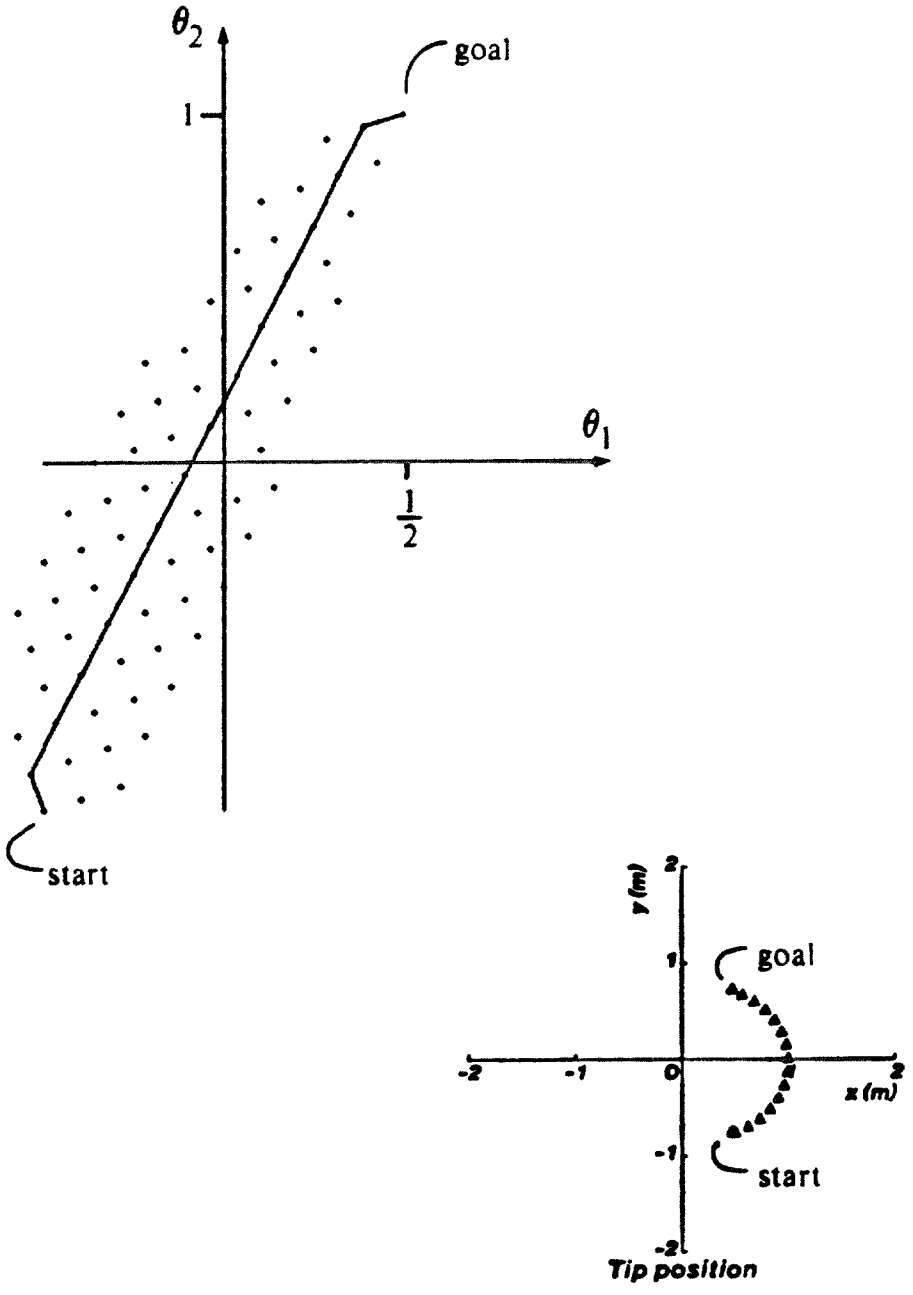


Figure 4: Moving from $(-0.5, -1.0)$ to $(0.5, 1.0)$: the path in joint and Cartesian spaces

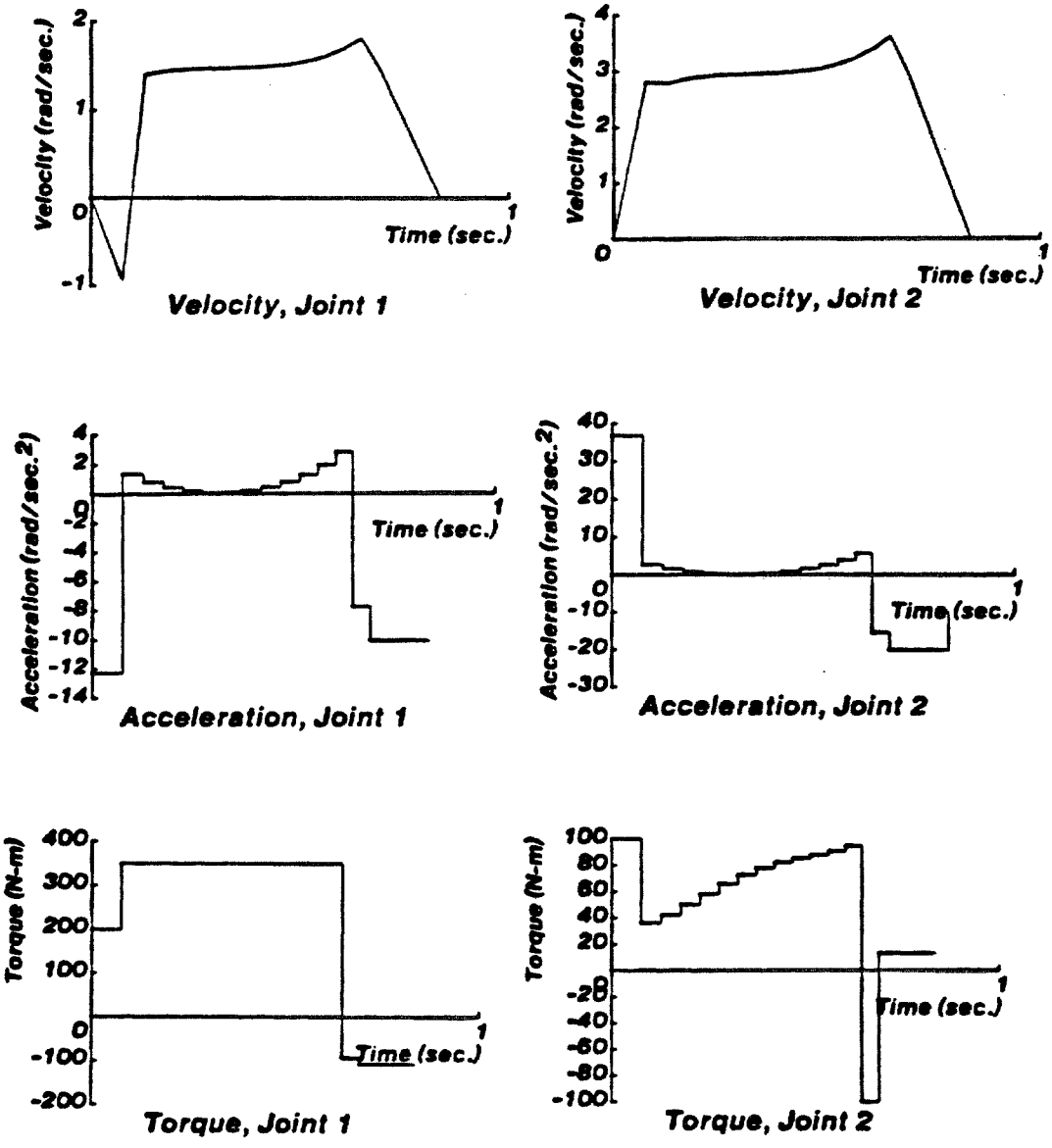
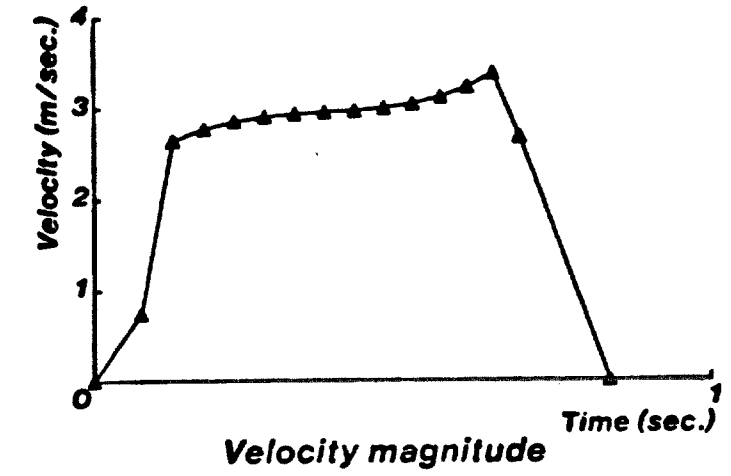


Figure 5: Moving from $(-0.5, -1.0)$ to $(0.5, 1.0)$: velocity, acceleration, and torque at the joints



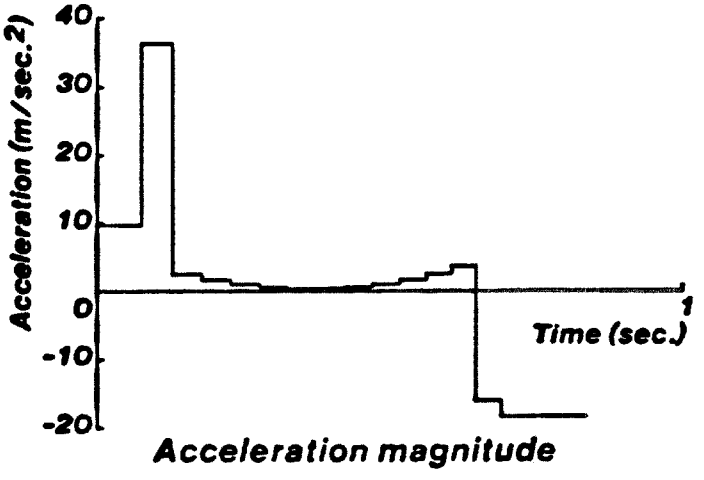


Figure 6: Moving from $(-0.5, -1.0)$ to $(0.5, 1.0)$: magnitude of tip velocity and acceleration in Cartesian space

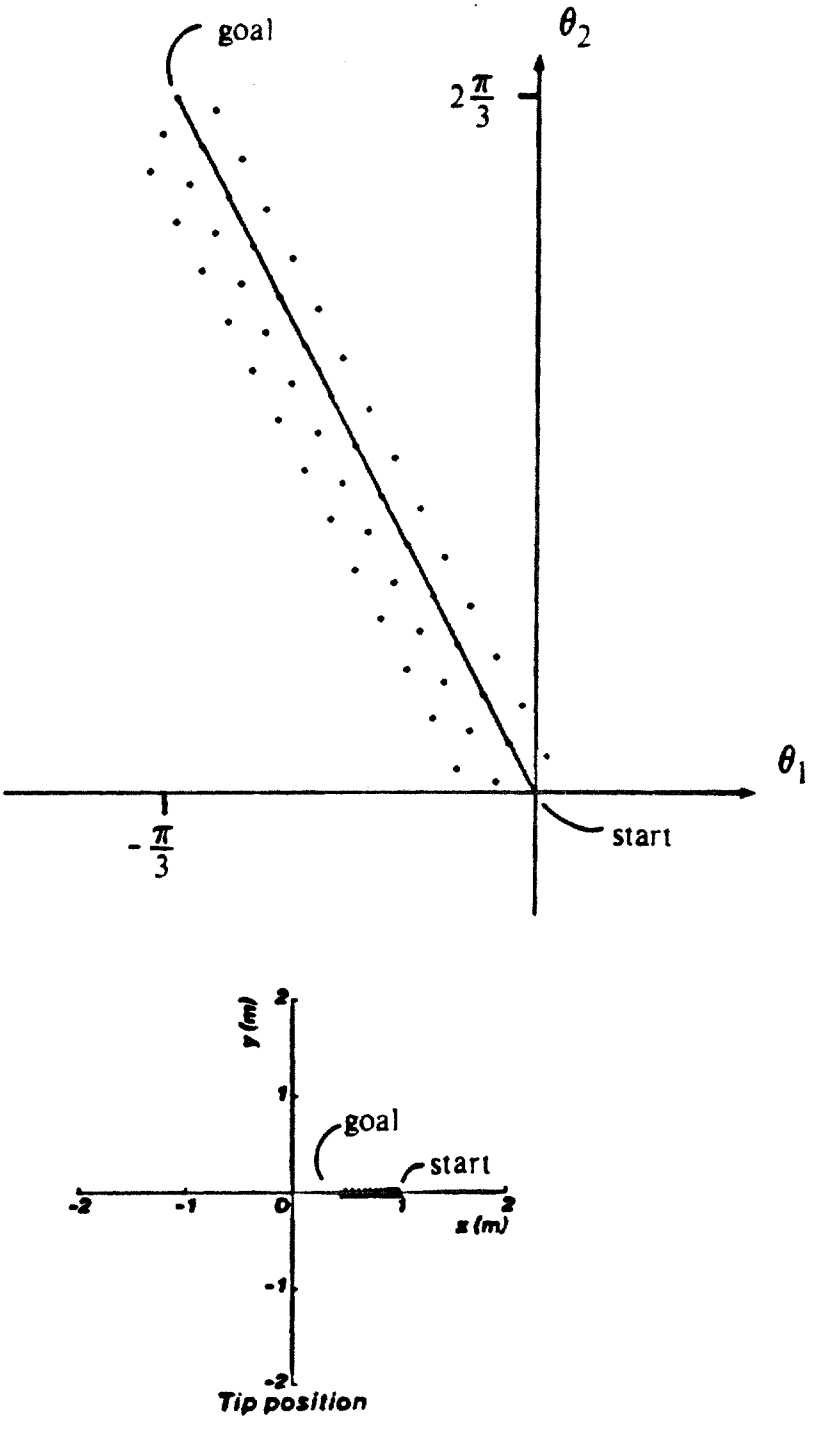


Figure 7: Moving from $(0.0, 0.0)$ to $(-\pi/3, 2\pi/3)$: the path in joint and Cartesian spaces

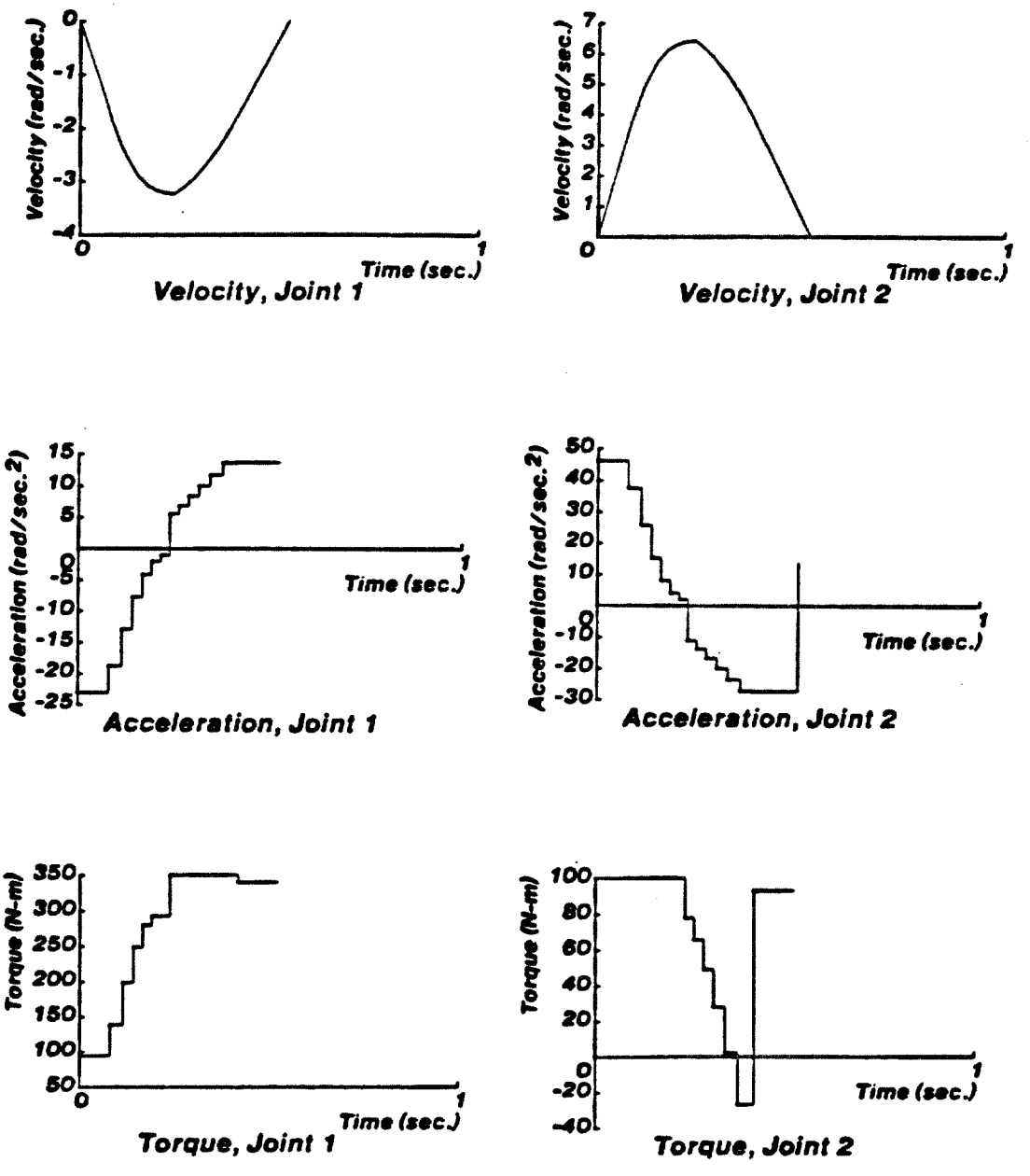
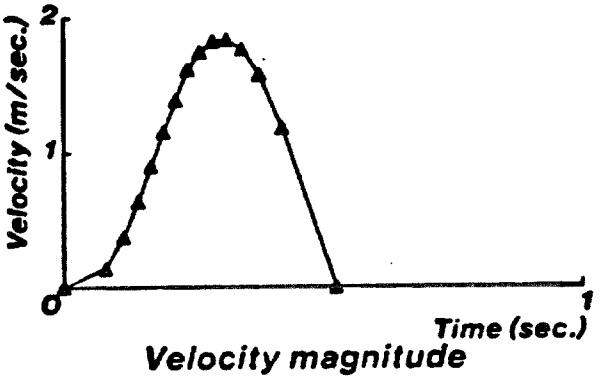


Figure 8: Moving from $(0.0, 0.0)$ to $(-\pi/3, 2\pi/3)$: velocity, acceleration, and torque at the joints



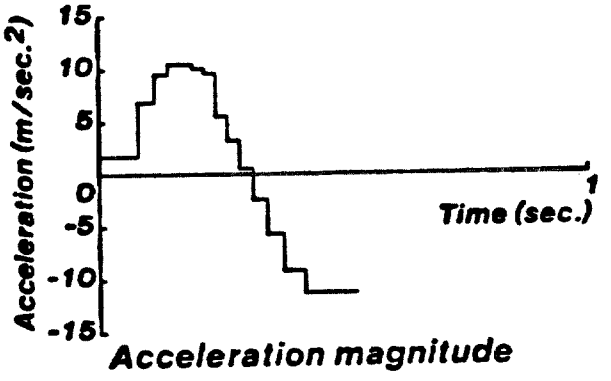


Figure 9: Moving from $(0.0, 0.0)$ to $(-\pi/3, 2\pi/3)$: magnitude of tip velocity and acceleration in Cartesian space

Bibliography

[1] Asada, H. "Dynamic Analysis and Design of Robot Manipulators Using Inertia Ellipsoids", *IEEE Computer Society 1st Intl. Conf. on Robotics*, Atlanta, Georgia, March 13-15, 1984, pp. 94-102.

[2] Bobrow, J.E. Optimal Control of Robotic Manipulators, Ph.D. Thesis, UCLA, 1982.

[3] Bobrow, J.E., Dubowsky, S., and Gibson, J.S. "On the Optimal Control of Robotic Manipulators with Actuator Constraints", *Proc. American Control Coference*, San Francisco, California, June 22-24, 1983, pp. 782-787.

[4] Brady, J.M., Hollerbach, J.M., Johnson, T.L., Lozano-Pérez, T., and Mason, M.T., eds., *Robot Motion: Planning and Control*, MIT Press, Cambridge, Massachusetts, 1982.

[5] Brown, M.L., Optimal Robot Path Planning via State Space Networks, M.S., Dept. Mechanical and Aerospace Eng., Princeton, June 1984.

[6] Dubowsky, S., and Shiller, Z. "Optimal Dynamic Trajectories for Robotic Manipulators", *Fifth CISM-IFToMM Symposium on Theory and Practice of Robots and Manipulators*, Udine, Italy, June 26-29, 1984, pp. 96-103.

[7] Hollerbach, J.M. "Dynamic Scaling of Manipulator Trajectories", MIT Artificial Intelligence Laboratory A.I. Memo 700, Jan. 1983a.

[8] Hollerbach, J.M. "Dynamic Scaling of Manipulator Trajectories", *Proc. of the American Control Conference*, San Francisco, California, June 22-24, 1983b, pp. 752-756.

[9] Hollerbach, J.M. "Dynamic Scaling of Manipulator Trajectories", *J. Dynamic Systems, Measurement, and Control*, **106**, 1984, pp. 102–106.

[10] Horn, B.K.P. "Kinematics, Statics, and Dynamics of Two-D Manipulators", MIT Artificial Intelligence Laboratory, Working Paper 99, June 1975.

[11] Kahn, M.E. "The Near-Minimum-Time Control of Open-Loop Articulated Kinematic Chains", Stanford Artificial Intelligence Laboratory, AIM 106, Dec. 1969.

[12] Kahn, M.E., and Roth, B. "The Near-Minimum-Time Control of Open-Loop Articulated Kinematic Chains", *J. Dynamic Systems, Measurement, and Control*, **93**, 1971, pp. 164–172.

[13] Lin, C.-S., Chang, P.-R., and Luh, J.Y.S. "Formulation and Optimization of Cubic Polynomial Trajectories for Industrial Robots", *IEEE Trans. Automatic Control*, **AC–28** (12), 1983, pp. 1066–1073.

[14] Lozano-Perez, T., "Task Planning," *Robot Motion: Planning and Control*, edited by Brady, Hollerbach, Johnson, Lozano-Perez, and Mason, MIT Press, 1982, ch. 6, pp. 473–498.

[15] Luh, J.Y.S., and Lin, C.S. "Optimum Path Planning for Mechanical Manipulators", *J. Dynamic Systems, Measurement, and Control*, **102**, 1981, pp. 142–151.

[16] Luh, J.Y.S., and Walker, M.W. "Minimum-Time Along the Path for a Mechanical Arm", *Proc. IEEE Conf. Decision and Control*, New Orleans, Louisiana, 1977, pp. 755–759.

[17] Sahar, G., Planning of Minimum-Time Trajectories for Robot Arms, Ocean Engineer Thesis, Dept. Ocean Engineering, MIT, September, 1984.

[18] Scheinman, V., and Roth, B. "On the Optimal Selection and Placement of Manipulators", *Fifth CISM-IFToMM Symposium on Theory and Practice of Robots and Manipulators*, Udine, Italy, June 26-29, 1984, pp. 25–32.

[19] Shin, K.G. and McKay, N.D. "An Efficient Robot Arm Control Under Geometric Path Constraints", *Proc. 22nd IEEE Conf. Decision and Control*, San Antonio, Texas, 1983, pp. 1449–1457.

[20] Winston, P.H. *Artificial Intelligence*, Second edition, Addison-Wesley Publishing Company, Reading, Massachusetts, 1984.