GLOBAL TIME OPTIMAL MOTIONS OF ROBOTIC MANIPULATORS IN THE PRESENCE OF OBSTACLES

Zvi Shiller * and Steven Dubowsky **

* Department of Mechanical, Aerospace and Nuclear Engineering
University of California Los Angeles, Los Angeles, CA 90024

** Department of Mechanical Engineering
Massachusetts Institute of Technology, Cambridge, MA 02139

ABSTRACT

A practical method to obtain the global time optimal motions of robotic manipulators is presented. This method takes into account the nonlinear manipulator dynamics, actuator constraints, joint limits, and obstacles. Previously developed methods for optimizing manipulator motions along given paths, and a local path optimization, are utilized. A set of best paths are obtained first in a global search over the manipulator work-space, using graph search and hierarchical pruning techniques. These paths are used as initial conditions for a continuous path optimization to yield the global optimal motion. Examples of optimized motions of a six DOF manipulator, operating in a three dimensional space with obstacles, are presented.

I. INTRODUCTION

High productivity of robotic manipulators performing tasks such as material handling and spot welding can be achieved if their motions be time optimal. Currently, manipulators are controlled far from optimally since most of the methods developed to date are not yet practical for realistic systems. Existing methods for the time optimization problem are either not yet computationally practical for complex nonlinear systems, such as articulated robotic manipulators[1-4], or they use simplified manipulator models to yield suboptimal solutions[5,6]. These methods have been demonstrated only for very simple two DOF systems, without addressing the problem of global optimality. The presence of obstacles in the manipulator's environment makes the task of obtaining global minimum-time motions even harder. Most methods for obstacle avoidance obtain obstacle free paths, without regard for the motion time[7-9].

This paper presents a practical procedure for the global time optimal motion planning of robotic manipulators with obstacles in their environment. The method is shown practical for realistic systems, such as six DOF manipulators operating in the presence of three dimensional obstacles. The optimization problem is decomposed into two sub-problems: *a*) optimizing the motion along a given path[10-13], and *b*) searching for the optimal path. This approach reduces the complexity of the overall problem by solving two simpler problems: the motion along a given path reduces to a one-dimensional optimization[10], and the path search is n dimensional (for n DOF manipulators). The path is searched hierarchically, using first a crude search in a tessellated space to obtain near-optimal paths which are then submitted to a continuous local optimization[14,15] to yield the global optimal solution.

The approach developed in this paper has been implemented on a microcomputer, μVAX II. Examples of time optimal paths of a six DOF manipulator, moving in a three dimensional space with obstacles, are presented. In these examples only the motions of the first three joints are optimized, while the orientation of the end-effector remains fixed. These examples demonstrate that the method is practical for realistic robotic systems, taking into account their full nonlinear dynamics, actuator and gripper constraints, obstacles, and joint limits.

II. GLOBAL SEARCH FOR TIME OPTIMAL PATHS

As mentioned earlier, the time optimization problem is transformed into a search in an n dimensional space, utilizing the time optimization along specified paths[12]. The constrained path optimization obtains efficiently the optimal velocity profile along a specified path, considering the manipulator dynamics, actuator saturation limits[10-11], and gripper and payload constraints[12-13]. Theoretically, by obtaining the time optimal motions along all possible paths between the end-points, we can find the global optimal one. However, because of the large number (infinite in a continuous space) of feasible paths between any two end-points, a hierarchical search is performed.

The objective of the hierarchical search is to reduce the initial set of all feasible paths between the end points to the global optimal one. Each path is evaluated by a lower bound estimate on its traveling time. Lower bound estimates, and not the actual optimal time for each path, are used in order to save computation time. Paths with a too high lower bound are pruned by upper bounds on the true optimal traveling time between the end-points, reducing the number of the remaining paths in the set. The hierarchical search is described below as a procedure.

2.1 The Hierarchical Search Procedure

Let

$\mathbf{P}^0$ = the set of all feasible paths between the end points.

$t(p)$ = the optimal time along a given path, $p \in \mathbf{P}^0$.

p_{opt} = the global optimal path

t_{opt} = $t(p_{opt})$ the global optimal time, where

$t_{opt} = \min \{t(p)\}$ for all $p \in \mathbf{P}^0$.

t_u = upper bound on the optimal time, where $t_u \geq t_{opt}$

$t^i(p)$ = the ith estimate for lower bound on the traveling time along a path, p, where

$t^i(p) \leq t^{i+1}(p)$ and

$\lim\{t^i(p)\} = t(p)$ as $i \rightarrow \infty$

Step 0: (start)

Set the upper bound $t_u = \infty$

Create the initial set of all paths $\mathbf{P}^0$

Set $i = 0$

Step 1:

Compute lower bound estimates $t^{i+1}(p)$ for $p \in \mathbf{P}^i$

Step 2:

For the best path p_{min} : $t^{i+1}(p_{min}) = \min\{ t^{i+1}(p) \}$

Compute the optimal time $t_{min} = t(p_{min})$

Update upper bound: $t_u = \min [t_u, t_{min}]$

Step 3:

Reduce set $\mathbf{P}^i$ to $\mathbf{P}^{i+1}$: $\{ p \mid t^{i+1}(p) \leq t_u \}$

Step 4:

If $t^{i+1}(p) < t(p)$

$i = i + 1$

go to step 1

Step 5:

Select the optimal path: p_{opt} : $t(p_{opt}) = \min\{ t(p) \}$;

for all $p \in \mathbf{P}^{i+1}$

Stop. //

This procedure reduces the initial set of paths, successively, by keeping only paths with lower bound estimates below an upper bound on the global optimal time. The convergence of the hierarchical search to the global optimal path is guaranteed under the conditions: 1) the estimates are true lower bounds, and 2) the initial set of paths must include the global optimal one. However, a practical implementation of this algorithm may loose some of its rigor, converging only to a near-optimal solution. Clearly, the finer the space representation, the closer the solution to the true optimum, but the heavier the computation load. In the rest of this paper, various aspects of the implementation of the hierarchical procedure are discussed.

III. IMPLEMENTATION

Since it is impossible to represent all feasible paths in the work-space, the objective of the hierarchical search is modified to find regions where local optimal paths may reside, using first a crude search in a tessellated space. A finer path optimization[14,15] is then used to yield the local optimal paths in the selected regions. The global optimal paths is selected from all local optimal solutions. We refer to the solution obtained here as *global optimal*, although we recognize the shortcomings of numerical optimization methods.

The success of the global search depends on the ability to explore all feasible regions in the work-space which may include the global optimal path. Hence, the main issue in the implementation of this algorithm is a proper representation of the work-space, and the initial selection of best paths. This issue is discussed below. Among other issues, discussed later, are obstacle avoidance, the lower bound estimates, and the upper bound for the optimal traveling time between the end-points.

3.1 Work-Space Representation

In the initial crude search, the work-space is represented by a grid. In such a representation the number of possible paths in the space is finite, and existing shortest paths algorithms[17,18] are applicable. A typical point on the grid represents, either, the vector of the position and the orientation of the end effector, relative to an inertial frame, $\underline{P}$, or the vector of the joint angles, $\underline{\theta}$. The method will be described only for a work-space implementation.

A single path in the grid is composed of a sequence of connected grid segments. Each grid segment represents the straight line connecting two adjacent grid points. The efficiency of the hierarchical search is a function of, both, the number of paths in the initial set which depends on the grid resolution, and the number of grid segments departing from each grid point. In a two-dimensional space, a typical grid point is connected to other eight nearest grid points. The choice of only eight neighboring grid points is arbitrary, and can be extended to points, not necessarily in the immediate vicinity of the center point. In three dimensions, the number of grid segments departing from a typical grid point to its nearest neighbors is 26.

A special graph representation is used which reduces the number of feasible paths between the end points by avoiding generating paths with small loops or sharp turns (turns of more than 45 degrees). This representation drastically reduces computation time, without limiting the ability to find the optimal path[16]. An additional state is added to the searched space in order to allow restricting the directions of departure as a function of the direction of arrival to a grid point. This representation transforms the work-space into a directed graph with the states $\{\underline{P}, d\}$, where d is the *direction* state. The graph is composed of nodes a_{ij} and edges e_{ij} ; i= 1,..n, j=1,..q, where n is the total number of grid points, and q is the number of different grid segments departing from a typical grid point. Each node has one incoming edge, and as many as permitted departing edges. Figure 1 shows possible arrivals and departures, to and from a typical grid point, in a two dimensional grid. In Figure 1, the incoming edge e_{12} from grid point 1 is connected to node a_{52} in grid point 5. This node is connected to only three departing edges in directions 1,2 and 3. These directions were chosen so as to avoid a sharp angle between the incoming and the departing edges.

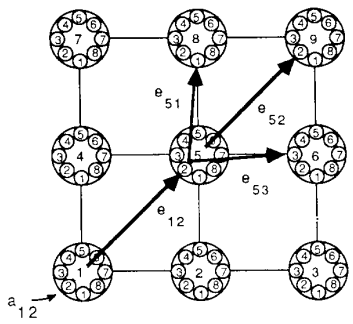

Figure 1: Typical Grid Points and Nodes Arrangements in 2D

3.2 Obstacle Avoidance

Obstacles are avoided by eliminating from the graph grid points which are not accessible by the manipulator tip because of interference between the manipulator's links and obstacles. These points form an *obstacle shadow* around each obstacle. Figure 2 shows a typical *obstacle-shadow* for an obstacle in the workspace of a two-link planar manipulator. The *obstacle shadow* is essentially a cartesian space representation of the *configuration-space obstacle*[9]. Work space representation has the disadvantage of considering only one kinematic configuration, leaving out the possibility of avoiding obstacles by switching from one configuration to the other. But it has the advantage of providing intuitive insights into the actual free space, allowing for interactive planning of work-cell layouts.

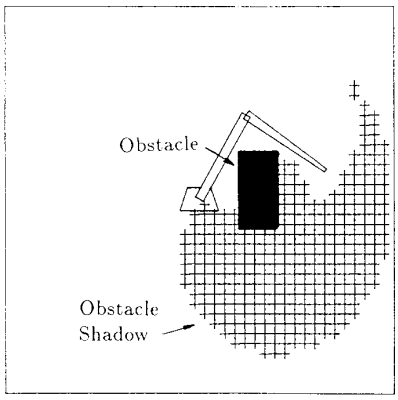

Figure 2: An Obstacle Shadow in the Workspace of a Two Link Manipulator

3.3 Lower Bound Estimates

Several lower bound estimates on the optimal traveling time along the path are used to quickly evaluate (test) paths for optimality. In the hierarchical search, the crude methods are used first when the number of paths is large, and the more accurate methods used last with a small number of paths. The methods selected in the implementation of the hierarchical search are described below.

3.3.1 Maximum Speed: The first lower bound estimate selects the first set of paths from the set of all possible paths between the end points. It is obtained by estimating the shortest traveling time along the path, assuming a constant speed. If this speed is the maximum operating speed of the manipulator tip, we obtain a true lower bound. The maximum speed can be determined from the actuator speed limits, or alternatively, it can be assigned heuristically to reflect an upper bound on the average traveling speed. Since initially the paths are composed of straight line segments, defined by the grid points, the lower bound, t^1, for the traveling time along a typical path is the summation of the costs of the individual segments:

$$t^1 = \sum_j \frac{d_j}{V_{max}} \qquad (1)$$

where d_j is the length of a typical grid segment, and V_{max} is the assigned maximum speed.

By assigning positive costs to all edges in the graph, it is now possible to obtain the set of best paths. The shortest path problem is relatively simple, and many methods with various computational efficiencies exist[18]. Obtaining the set of K shortest paths, however, is more difficult. A method by Dreyfus[17] was chosen. After the first set of paths is selected, each path is smoothed with cubic splines, using the grid points as the control points[19]. The next tests for lower bounds estimate the traveling time along smooth paths.

3.3.2 Velocity Limit Curve: The next test is composed of the travelling time along the velocity limit curve. The velocity limit is the maximum permissible speed along a given path which can be sustained by the manipulator, given actuator constraints, system's dynamic characteristics, and the path[10]. The second lower bound t^2 is obtained by the integral:

$$t^2 = \int_{S_0}^{S_f} \frac{1}{\dot{S}_m} dS \qquad (2)$$

where $\dot{S}_m$ is the velocity limit curve[11], and S is the distance along the path. Since the actual velocity profile of the manipulator tip along the path should not exceed the limit curve, the time obtained in Equation (2) is a lower bound. This test requires the computation of manipulator dynamics, and is computationally more expensive than the previous one, but it reflects better the actual manipulator dynamic behavior.

3.3.3 Max. Acceleration, Velocity Limit, Max. Deceleration: The third test considers the acceleration and deceleration from the initial and final points, without exceeding the velocity limit curve. It is obtained by the summation:

$$t^3 = \int_{S_0}^{S_1} \frac{1}{\dot{S}_a} dS + \int_{S_1}^{S_2} \frac{1}{\dot{S}_m} dS + \int_{S_2}^{S_f} \frac{1}{\dot{S}_d} dS \qquad (3)$$

where $\dot{S}_a$ and $\dot{S}_d$ are the speeds along the path during acceleration and deceleration, respectively, and S_1 and S_2 are the points along the paths where $\dot{S}_a$ and $\dot{S}_d$ exceed the velocity limit $\dot{S}_m$, respectively. This test is a better estimate then the previous one, but still a lower bound of the true optimal time. The computation required for this test is slightly higher than the velocity limit test.

The relative importance of each method depends on the actual system. For simple systems, the first test may provide an accurate measure for optimality, without the need for the other tests. For the manipulator used here, the last test (acceleration, velocity limit and deceleration) provides the best estimate for optimality. It has been shown that the acceleration and deceleration near the end points play an important role in determining the traveling time along a given path[20,21]. After each test, the paths with too high a lower bound are pruned by an upper bound on the optimal time.

3.4 Upper Bounds

The upper bound on the optimal traveling time between given end-points is obtained by computing the true optimal traveling time along some smooth path between these points, using the specified path optimization[12]. Clearly, the closer the path to the optimal, the lower the upper bound. The upper bound is updated by obtaining the true optimal time for several best paths after each estimate (test). The upper bound is computed only for a few paths, because this method is computationally more expensive than the other lower bound tests.

3.5 Filtering

Since only one path is needed in the neighborhood of each local optimum for the path optimization, the paths remaining after the last pruning process are filtered so as to retain the best path in every region. The main justification for the filtering is the assumption that two similar paths, $\underline{x}_1$ and $\underline{x}_2$, yield similar traveling times, or

$$D(\underline{x}_1,\underline{x}_2) < \varepsilon \implies |t(\underline{x}_1) - t(\underline{x}_2)| < \delta;\ \delta, \varepsilon \ll 1 \qquad (4)$$

where $t(\cdot)$ is the traveling time along a given path, and $D(\cdot)$ is the distance between two paths, defined by:

$$D(\underline{x}_1,\underline{x}_2) = \max \{ | (\underline{x}_1(u) - \underline{x}_2(u) | \} \qquad u = [0,1] \qquad (5)$$

where $\underline{x}_i$ is a vector composed of the coordinates, the slope and curvature at a point on path i, and u is a normalized path distance. The paths are filtered by eliminating all paths within a short distance from the best path in each region.

IV. EXAMPLES

The examples presented below demonstrate the method for the global time optimal paths on a simple two link manipulator, and a six DOF manipulator operating in a three dimensional workspace with obstacles. These examples were computed on a microcomputer, μVAX II.

4.1 Two Dimensional Optimization With Obstacles

Figure 3 shows a two link manipulator at the end points, and four obstacles in the manipulator work space. The manipulator's parameters are given in Table 1. Also shown in Figure 3 are the *obstacle shadows*, assuming that the links and the obstacles are in the same plane. The grid points inside *obstacle shadows* are marked by small crosses; points out of reach or points of singularity are marked by x's. Figure 4 shows 238 best feasible paths between the end points, produced by the initial graph search. Since many of the paths share same path segments, it is impossible to distinguish between individual paths in Figure 4. Figure 5 shows the 57 paths remained after the hierarchical search, before filtering, with the times ranging from 1.47 to 2.07 sec. After filtering, only one path remained, with the traveling time of 1.471 seconds. In this case, there is only one local optimal region since the other possible local optima are not feasible because of the presence of obstacles. Submitting this path to the local optimization yielded the global optimal path with the optimal time of 1.416 seconds. Figure 6 shows the manipulator at various points along the optimal path. The computation time for the hierarchical pruning was about 3 minutes of CPU time, and for the local optimizations was about 10 minutes.

Table 1 Manipulator's Parameters

$l_1 = 0.5$ m	$L_1 = 1.0$ m	$m_1 = 1$ kg	$I_1 = 0.2$ Kg m^2	$T_1 = 10$ N m
$l_2 = 0.5$ m	$L_2 = 1.0$ m	$m_2 = 1$ kg	$I_2 = 0.2$ Kg m^2	$T_2 = 10$ N m

4.2 Three Dimensional Paths for a Six DOF Manipulator

The following results were obtained by optimizing the motion of a six DOF manipulator, moving in a three dimensional space. The six dimensional manipulator dynamics are considered, but only the motion of the first three joints is optimized. During the entire move, the last three joints remain at a fixed orientation relative to the inertial frame. Even though the motion of the wrist is not optimized, it cannot be ignored, since one of the small actuators at the wrist may be limiting the speed along the path. The grid representation of the work space consisted of a 10x10x10 grid.

Figures 7 shows top and perspective views of the optimal path of a six DOF manipulator moving in a three dimensional space, from work station A to work station B, while avoiding obstacles C and D, with the traveling time of 0.51 seconds. It is important to note, that the optimal path with obstacles depends on the penalty function chosen to represent the obstacles[14]. It is possible to make the path stay away from an obstacle by selecting the appropriate weighting factors in the penalty function. The computation time for these cases were in the order of 2 hours.

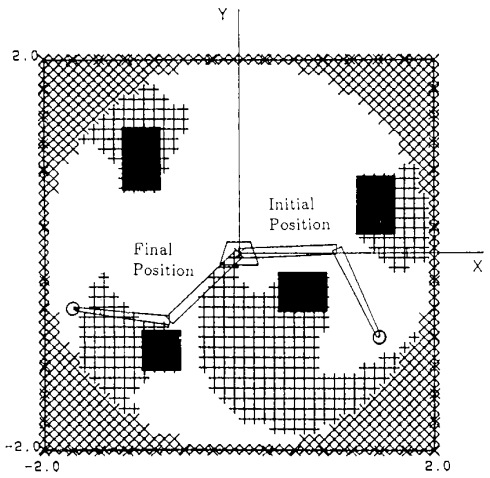

Figure 3: A Two-Link Manipulator and Obstacles

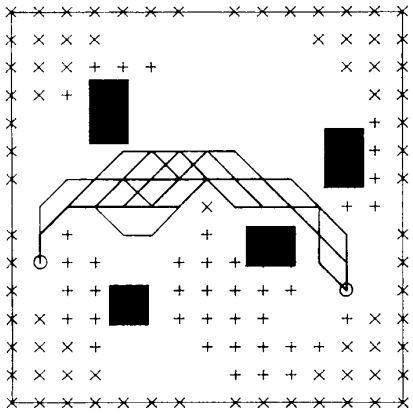

Figure 4: Best Paths in the Tessellated Space

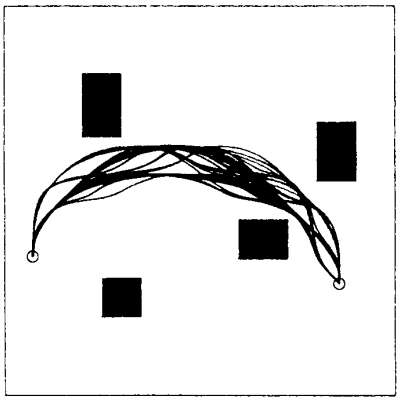

Figure 5: Best Smoothed Paths After Hierarchical Pruning

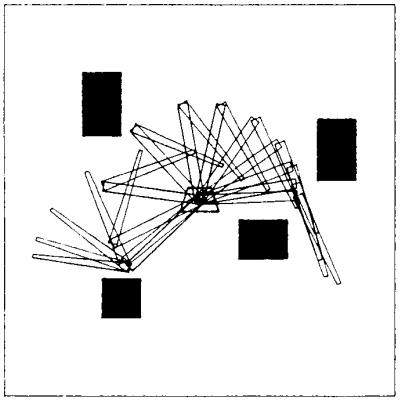

Figure 6: The Manipulator at Various Points Along the Time Optimal Path

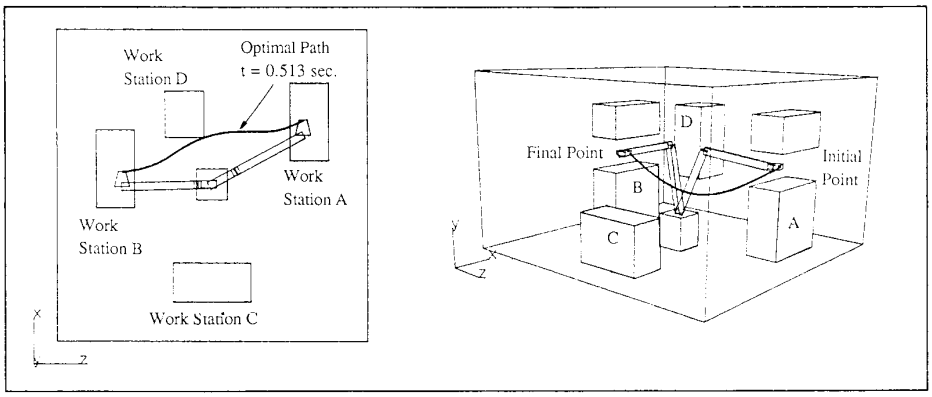

Figure 7: Time Optimal Path for a Six DOF Manipulator in a 3D Space with Obstacles

V. CONCLUSIONS

A method that obtains practically the global optimal motion for a manipulator, considering its dynamics, actuator constraints, joint limits, and obstacles, has been presented in this paper. This method consists of a hierarchical search for the best path in a tessellated space, which is used as the initial conditions for a local path optimization to yield the global optimal path. Implemented on a μVAX II microcomputer, the algorithm is demonstrated for a two link manipulator, and shown computationally practical for off-line path planning of realistic systems, such as a six DOF manipulator, operating in a three dimensional space with obstacles.

VI. ACKNOWLEDGEMTNS

This research was done at the Massachusetts Institute of Technology, supported by the Automation Branch of NASA Langley research Center under Grant NAG-1-489.

VII. REFERENCES

1. Geering, H., Guzzella, L., Hepner, S.A.R., and Onder, C.H., "Time-Optimal Motions of Robots in Assembly Tasks", *IEEE Transactions on Automatic Control*, Vol. AC-31, No. 6, June 1986, pp. 512-518.
2. Weinrab, A. and Bryson, A.E., "Optimal Control of Systems With Hard Control Bounds," Proc. 1985 ACC, Boston, MA., pp. 1248-1252, June 1985.
3. Kiriazov, P. and Marinov, P., "A Method for Time-Optimal Control of Dynamically Constrained Manipulators," in *Theory and Practice of Robots and Manipulators*, pp. 169-178, Kogan Page, London 1985, MIT Press 1985.
4. Sahar, G. and Hollerbach, J.M., "Planning of Minimum-Time Trajectories for Robot Arms," Proc. of the IEEE Intr'l Conf. on Robotics and Automation, pp. 751-758, St. Louis, Mo., March, 1985.
5. Kahn M. E. and Roth B., "The Near-Minimum-Time Control of Open-Loop Articulated Kinematic Chains," *J. of Dynamic Systems, Measurement and Control*, Vol. 93, No. 3, Sept. 1971, pp. 164-172.
6. Wen, J. T. and Desrochers, A.A., "Sub-Optimal Control Strategies for Robotic Manipulators," Proc. of IEEE Intr'l Conf. on Robotics and Automation, San-Francisco, April 1986, pp. 402-406.
7. Brooks, R. A., "Planning Collision Free Motions for Pick and Place Operations," *Intr'l J. of Robotics Research*, Vol. 2, No. 4, 1983.
8. Khatib, O., "Real-Time Obstacle Avoidance for Manipulators and Mobile Robots", *Intr'l J. of Robotics Research*, Vol. 5, No. 1, Spring 1986.
9. Lozano-Perez, T., "A Simple Motion Planning Algorithm for General Robot Manipulators," *IEEE J. of Robotics and Automation*, Vol. RA-3, No. 3, pp. 224-238, June 1987.
10. Bobrow, J. E., Dubowsky, S., and Gibson, J. S., "Time-Optimal Control of Robotic Manipulators," *Intr'l J. of Robotics Research*, Vol.4, No. 3, 1985.
11. Dubowsky, S. and Shiller, Z., "Optimal Dynamic Trajectories for Robotic Manipulators," in *Theory and Practice of Robots and Manipulators*, pp. 133-143, Kogan Page, London 1985, MIT Press 1985.
12. Shiller, Z. and Dubowsky, S., "On the Optimal Control of Robotic Manipulators with Actuator and End-Effector Constraints," Proc. of IEEE Intr'l Conference on Robotics and Automation, pp. 614-620, St. Louis, Mo., March 1985.
13. Shiller, Z. and Dubowsky, S.,"Time Optimal Path Planning for Robotic Manipulators in the Presence of Obstacles, with Actuator, Gripper and Payload Constraints," Submitted to the *Intr'l J. of Robotics Research*, Aug. 1987.
14. Dubowsky, S., Norris, M.A. and Shiller, Z., "Time Optimal Trajectory Planning for Robotic Manipulators with Obstacle Avoidance: A CAD Approach", Proceedings of IEEE International Conference on Robotics and Automation, pp. 1906-1912 San Francisco CA., April 1986.
15. Dubowsky, S., Norris, M.A. and Shiller, Z., "Time Optimal Robotic Manipulator Task Planning", Proceedings of the Six CISM-IFToMM Symposium on Theory and Practice of Robots and Manipulators, Sept. 1986, Cracow, Poland.
16. Shiller, Z., "Time Optimal Motion Planning for Robotic Manipulators," Doctoral Thesis, Department of Mechanical Eng., Massachusetts Institute of Technology, Cambridge MA, June 1987.
17. Lawler, E. L., 1976, *Combinatorial Optimization: Networks and Matroids*, Holt, Rinehart and Winston, 1976. pp. 98-100.
18. Pearl, J., 1984, *Heuristics: Intelligent Search Strategies for Computer Problem Solving*, Addison-Wesely, 1984.
19. Mortenson M. E., 1985, *Geometric Modeling*, John Wiley & Sons 1985.
20. Shiller, Z. and Dubowsky, S., "The Acceleration Map and its Use in Minimum Time Motions Planning of Robotic Manipulators," Proc. 1987 ASME Int'l. Computers in Engineering Conference, New-York, NY, August 1987.
21. Shiller, Z. and Dubowsky, S., "Time Optimal Paths and Acceleration Lines of Robotic Manipulators", 26th IEEE Conf. on Decision and Control, Los Angeles, Dec, 1987.