Robotica (1997) volume 15, pp 31–40. © 1997 Cambridge University Press

Optimal Robot Motion Planning and Work-Cell Layout Design

Zvi Shiller

Department of Mechanical and Aerospace Engineering, University of California Los Angeles, Los Angeles, Ca 90095-1597 (USA)

SUMMARY

This paper describes an interactive software system, developed at the Robotics and Automation Laboratory at UCLA to demonstrate innovative approaches to off-line robot programming and work-cell layout design. The software computes the time-optimal motions along specified paths, local optimal paths around an initial guess, and the global optimal path between given end-points. It considers the full robot dynamics, actuator constraints, on the payload acceleration or the gripping force, and any number of polygonal obstacles of any shape. The graphic displays provide a useful tool for interactive motion planning and workcell design.

KEYWORDS: Motion planning; Robots; Work-cell design; Software systems.

1. INTRODUCTION

Motion planning is an essential step in robot applications on the factory floor. It consists of selecting a nominal trajectory for the joint controllers that avoids obstacles and possibly minimizes distance, joint rates, joint torques, or motion time. The need for a nominal trajectory stems from the nonlinear and coupled robot dynamics and obstacles, which are too complex for current joint controllers.

Early research in this field focused on *kinematic* motion planning, emphasizing obstacle avoidance (see for example the paper by Lozano-Perez[1]), whereas more recent research has focused on *dynamic* motion planning, concerning robot dynamics, actuator constraints, and motion time.[2,3] Minimizing motion time can significantly shorten cycle times, improve machine utilization, and thus make automation affordable in applications for which throughput and cost effectiveness is of major concern.

The first attempt to compute time-optimal robot motions dates back to the work by Kahn and Roth.[4] Their approach was based on the Pontryagin's Maximum Principle, that although theoretically elegant, was computationally impractical for robotic manipulators with their full nonlinear dynamics. Practical approaches to this problem were facilitated by the important work by Bobrow,[5] who efficiently solved the problem of computing the time-optimal motions along specified paths. This method was shown computationally practical for full six DOF manipulators,[6] and was later extended to consider gripper and payload constraints,[3] and control singularities.[7,8] This method was then used to solve the more difficult point to point problem, first addressing the local,[3] and then the global[9] optimal problem.

The generation of a good initial guess for the local optimization is still an open problem. One promising approach is based on the *acceleration lines,* shown to produce paths to within 5% of the time-optimal path.[10] It is based on the observation that time-optimal paths are generally nonsingular (the optimal control is bang-bang), and thus maximize the acceleration or deceleration near the end-points, or equivalently, are tangent to the *acceleration lines* near the end points.

The CAD software described in this paper combines the path optimizations at the various levels (specified path, local and global optimizations) with graphics interfaces to facilitate interactive robot programming and workcell layout design. The graphic interfaces include the *acceleration lines,* and the representation of obstacles by their *obstacle shadows.* Obstacle shadows are used to interactively plan collision-free paths in the workspace. User inputs include the geometry of the cell layout (size and location of rectangular obstacles), robot dynamic parameters (link length, mass and inertia), path parameters (the control points of a B spline), or the end points (for path optimization). The output consists of the optimal trajectory (path and speed) that minimizes motion time and avoids obstacles. This trajectory can be used as a nominal trajectory for the on-line controllers. In the current implementation a specific robot model is used, however, except for very few subroutines, the program is robot independent. Work-cell design is currently done interactively by placing workstations so as to not obstruct the optimal paths between other workstations. The interference with an optimal path, approximated by the acceleration lines, is detected using obstacle shadows, as is demonstrated in an example in this paper.

This paper briefly presents the theory behind the main computational modules of the interactive software. Detailed discussions are presented in the related papers.[3,7–10]

2. INTERACTIVE ROBOT MOTION OPTIMIZATION (INOPT)

The interactive motion planning software consists of the following main modules:

- OPTARM I: time optimal motions along a specified path
- OPTARM II: local optimal path between given end points

https://doi.org/10.1017/S0263574797000052 Published online by Cambridge University Press

- Optarm III: global optimal path between given end points
- ACCMAP: graphic display of robot acceleration capabilities
- ACCLINES: graphic display of acceleration lines
- OBSHAD: graphic display of obstacle shadows

2.1 OPTARM I: Time optimal motions along specified paths

The OPTARM I module computes the time optimal velocity profile of the manipulator along a specified path. It is based on an algorithm, first developed by the Bobrow[2] and later modified by Shiller and Lu,[7] which considers manipulator dynamics and actuator constraints.

The algorithm is based on parameterizing the specified path either in the joint or work-space coordinates, as a function of arc length, s:

$$q = f(s) \tag{1}$$

where $f(\cdot)$ represents the transformation from path coordinates to the joint space. Differentiating (1) twice with respect to time yields the joint velocities and accelerations as functions of the velocity $\dot{s}$ and acceleration $\ddot{s}$ along the path:

$$\dot{q} = f_s \dot{s} \tag{2}$$

$$\ddot{q} = f_s \ddot{s} + f_{ss} \dot{s}^2 \tag{3}$$

where f_s and f_{ss} are the first and second derivatives of $f(s)$ with respect to s. Using the kinematic transformations in (3), the equations of motion, which are originally written in the form:

$$M\ddot{q} + \dot{q}^T C\dot{q} + G = T \tag{4}$$

are transformed to the equivalent equations in terms $\dot{s}$ and $\ddot{s}$:

$$m(q)\ddot{s} + b(q)\dot{s}^2 + G(q) = T \tag{5}$$

where

$$m = Mf_s$$

$$b = Mf_{ss} + f_s^T C f_s$$

In (4), q, $\dot{q}$, and $\ddot{q}$ are the $n \times 1$ vectors of joint displacements, velocities and accelerations, respectively, for an n dimensional system, $M(q)$ is the $n \times n$ inertia matrix, $C(q)$ is the $n \times n \times n$ array of the coefficients of the centrifugal and Coriolis forces, $G(q)$ is the $n \times 1$ vector of the potential forces, and T is the vector of actuator efforts. Actuator saturation limits can be any functions of joint positions and velocities:

$$T_{imin}(q, \dot{q}) \le T_i \le T_{imax}(q, \dot{q}); \; i = 1, \ldots, n \tag{6}$$

For simplicity, the actuator bounds are assumed constant.

For a given s and $\dot{s}$, (5) defines a unique mapping between T and $\ddot{s}$, assuming m_i and b_i (the ith elements of the vectors m and b) are not simultaneously zero for any $i = 1, \ldots, n$. We can therefore use s and $\dot{s}$ as the two independent states, and $\ddot{s}$ as the independent control variable. The problem of minimizing the motion time of the manipulator along the path is then reduced to

$$\min_{\ddot{s}} J = \int_0^{t_f} 1 \, dt \tag{7}$$

subject to the scalar state inequality constraint

$$\dot{s} \le \dot{s}_{max}(s) \tag{8}$$

where[7]

$$\dot{s}^2_{max} = \min_{ij} \left(\max_{T_i T_j} \left(\frac{m_j(T_i - G_i) m_i(T_j - G_j)}{m_j b_i - m_i b_j} \right) \right); \; i, j = 1, \ldots, n \tag{9}$$

and to the state dependent control constraints

$$\ddot{s}_d \le \ddot{s} \le \ddot{s}_a \tag{10}$$

where

$$\ddot{s}_d = \max_{\dot{s}} \left(\frac{T_{imin} - G_i - b_i \dot{s}^2)}{m_i} \right); \; i = 1, \ldots, n \tag{11}$$

$$\ddot{s}_a = \min_{\dot{s}} \left(\frac{T_{imax} - G_i - b_i \dot{s}^2}{m_i} \right); \; i = 1, \ldots, n \tag{12}$$

assuming $m_i > 0$ for all $i = 1, \ldots, n$. The points along the path where $m_i = 0$ are singular points.[7] These are handled as state inequality constraints, as discussed below.

The maximum velocity $\dot{s}(s)_m$ represents the velocity value above which (10) cannot be satisfied with the given actuator constraints. This is also the value at which $\ddot{s}_d(\dot{s}_{max}(s)) = \ddot{s}_a(\dot{s}_{max}(s))$, excluding singular points. Plotting $\dot{s}_m$ in the phase plane ($\dot{s}$ vs s) forms the velocity limit curve which serves as an upper bound for the actual velocity profile. The time optimal motion thus consists of the highest reachable speeds which do not cross the limit curve. It can be shown that with the absence of singular points, the time optimal trajectory consists of extremal arcs, i.e. it is bang-bang in the acceleration along the path.[7] The simplest optimal trajectory consists therefore of two arcs, with one switch from maximum acceleration to maximum deceleration. It is computed by integrating the maximum acceleration from the initial to the final point, and integrating the maximum deceleration from the final point until it intersects the maximum acceleration trajectory. A trajectory with more than one switch must be tangent to the limit curve[11] at least at one

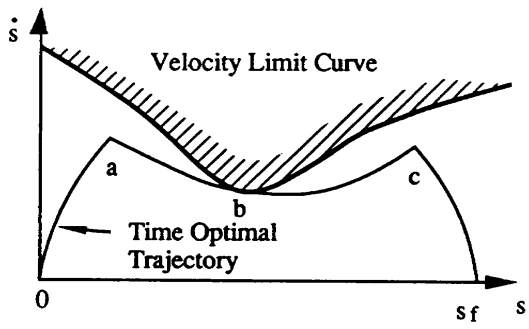


Fig. 1. A typical velocity limit curve and a time optimal trajectory.

https://doi.org/10.1017/S0263574797000052 Published online by Cambridge University Press

point, which is a switching point from maximum deceleration to maximum acceleration. A typical trajectory with three switches and four extremal arcs is shown schematically in Figure 1. The trajectory shown accelerates from the initial point to point *a,* then decelerates to the tangency point *b,* accelerates again to point *c,* and finally decelerates to the final point. Any other trajectory is either sub-optimal, if it attains lower speeds, or is infeasible if it crosses the limit curve.

The tangency point is a unique point at which the following tangency condition holds[7]

$$\ddot{s}(\dot{s}_{max}(s)) = \frac{d\dot{s}_{max}(s)}{ds}\dot{s}_{max}(s) \tag{13}$$

Other points at which the optimal trajectory can touch the limit curve are points where the limit curve is first-order indifferentiable. These points are potential switching points between extremal deceleration and acceleration arcs.[7] They include discontinuity points, where the path is either discontinuous or indifferentiable, or singular points, where $m_i(s) = 0$ for some $i = 1, \ldots, n$.

At singular points,

$$\ddot{s}_d(\dot{s}_{max}(s)) \neq \ddot{s}_a(\dot{s}_{max}(s)). \tag{14}$$

If the acceleration on each side of the singular point is interior to the acceleration bounds (10), then it becomes an *active* singular point.[7] Singular points are the only points along the time optimal trajectory at which the acceleration is not maximized nor minimized, but rather determined by the state inequality constraint (8). These are also the only points where the state inequality constraint is independently active, since at all other points the satisfaction of the control inequality constraints is sufficient to determine the optimal control. There are cases where the singular points extend to a singular arc along which the trajectory slides along the velocity limit curve. The algorithm implemented in OPTARM I takes singular points into account, and is therefore robust to robot and path parameters.

Recognizing singular points is essential since attempting to integrate the maximum acceleration or deceleration at such points may result in serious numerical difficulties due to chatter.[7] It was shown that singular points may occur only if the set of admissible control is not strictly convex.[8] One way to avoid singular points, and thus simplify the computation of the optimal trajectories, is to convexify the set of admissible control by considering the largest hyper-ellipsoid enclosed by the set of admissible controls.[8]

In addition to actuator constraints, this optimization can consider any constraint that can be expressed in terms of end-effector speed and acceleration. The treatment of constraints such as the maximum dynamic forces that the payload can tolerate, and the maximum speeds which a manipulator can sustain without losing its grasp of the payload, has been presented previously[3] and implemented in the OPTARM I module. Figure 2 shows a typical time optimal trajectory computed by OPTARM I.

The output of this module includes the optimal traveling time along the path, optimal velocity profile, and actuator torques. Figure 3 to 5 show typical computer displays of the workcell, the manipulator and the path (Figure 3), its time-optimal velocity profile (Figure 4), and the optimal actuator torques (Figure 5). Typical computation times for a six DOF manipulator are less than one second on a Silicon Graphics IRIS 4D70 workstation. This module can be replaced with an algorithm that minimizes a time-energy cost function to produce smooth trajectories. Time-energy optimal

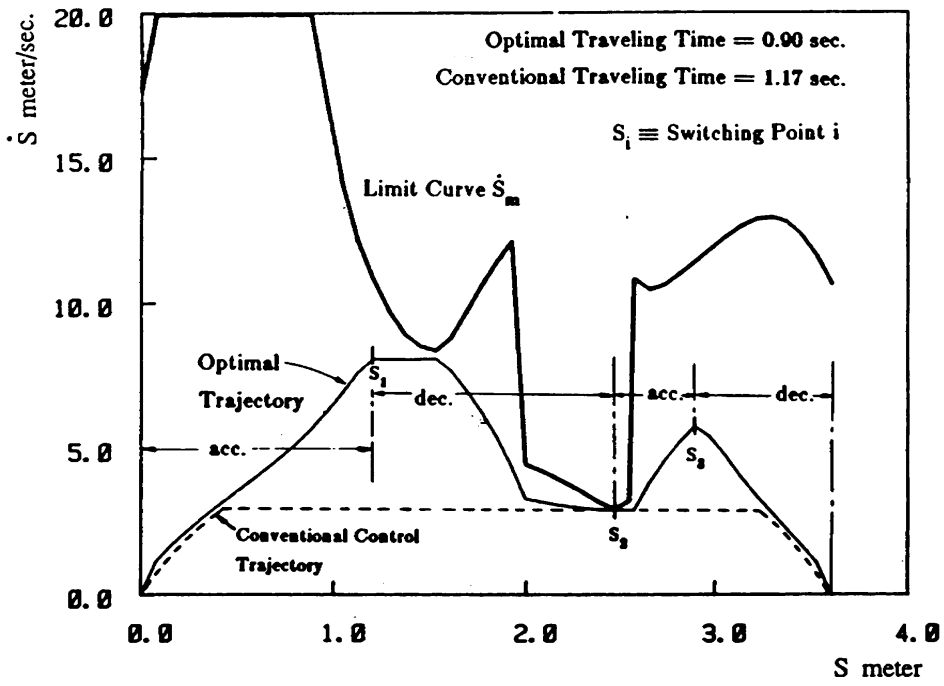


Fig. 2. A typical time optimal trajectory computed by OPTARM I.

https://doi.org/10.1017/S0263574797000052 Published online by Cambridge University Press

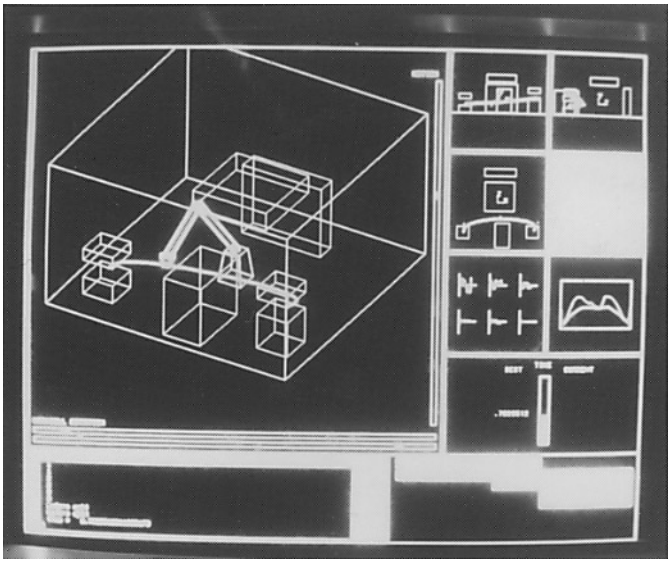


Fig. 3. The workcell.

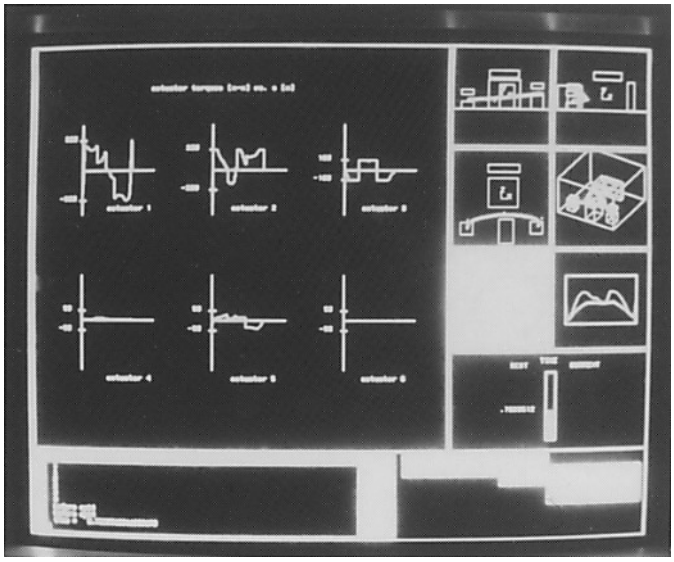


Fig. 5. Optimal actuator torques.

trajectories were shown to yield better tracking accuracy than the typically nonsmooth time-optimal trajectories.[12]

2.2 OPTARM II: Local path optimization

The interactively selected path can be further improved by a path optimization which automatically computes the time optimal and obstacle-free path between the end points.[3] The structure of the path optimization module, OPTARM II, is depicted in Figure 6.

To practically compute the local optimal path, the optimization problem is formulated as an unconstrained parameter optimization over path parameters by minimizing the cost function

$$\min_{\mathbf{a}} J = t(\mathbf{a}) + P(\mathbf{a}) \tag{15}$$

where $t(\cdot)$ is the optimal traveling time along the path, $\mathbf{a}$ is the vector of path parameters (the control points of the B spline), and $P(\cdot)$ is a penalty function on the distance of the manipulator from obstacles:

$$P(\mathbf{a}) = \sum \frac{w_i}{d_i^2}(\mathbf{a}) \tag{16}$$

where d_i is the minimum distance between the manipulator and the ith obstacle, and w_i is the weighting factor for this obstacle. This formulation greatly simplifies the optimization problem since it avoids the more traditional variational approach, which is based on satisfying necessary conditions[13] of optimality. Compared to the variational approach, ours requires no co-states, and is therefore formulated in a reduced state-space. Consequently, this approach is not as sensitive to the initial guess. In addition, state inequality constraints, such as imposed by the obstacles, are easy to consider, and since the boundary conditions can always be satisfied, the optimization can be terminated at any time for a feasible (sub-optimal) solution. It is important to note that this approach produces a solution that is only an approximation of the true optimum, which depends on the choice of the path variables, $\mathbf{a}$, and the weighting factors w_i.

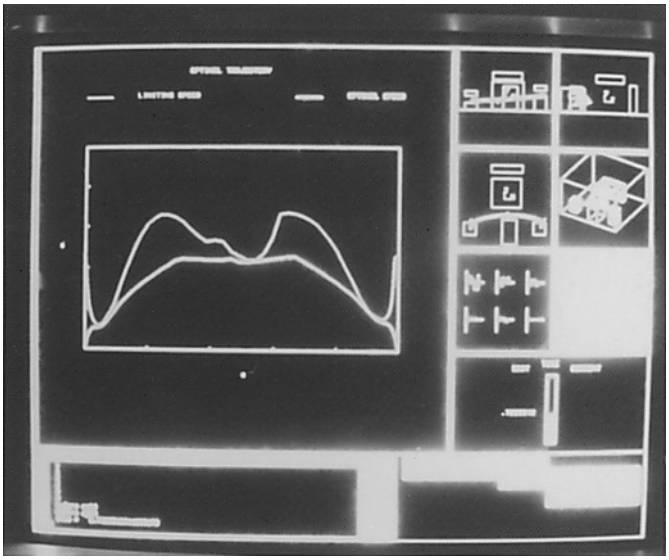


Fig. 4. Time-optimal velocity profile.

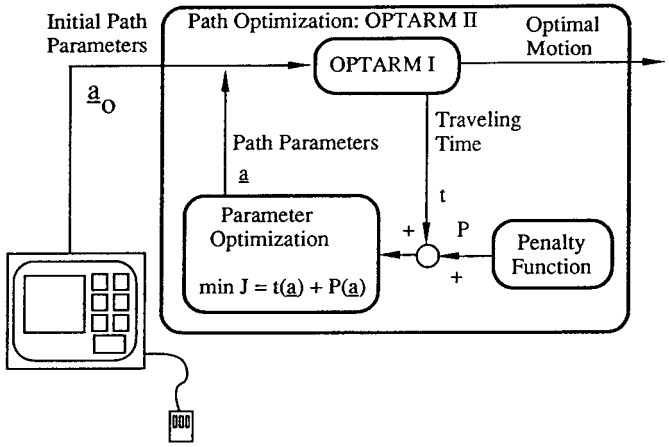


Fig. 6. Path optimization.

https://doi.org/10.1017/S0263574797000052 Published online by Cambridge University Press

In OPTARM II, a pattern search is used to optimize the cost function (15). The pattern search is a zero order search, based on direct evaluations of the cost function. The pattern search does not require the gradient, which is difficult to compute for this problem since the cost function computed by OPTARM I is not always smooth over the entire parameter space due to the discrete nature of the path representation.

Starting with an initial guess of the parameters, $\mathbf{a}_0$, the optimization algorithm iterates on path parameters so as to minimize the cost function. The user can monitor the optimization on the graphic display and interrupt it any time to modify the current best path, or to test another initial guess. Computation time for a six DOF manipulator can vary from 10 to 30 min on IRIS-4D70 (much faster on newer machines), depending on the number of obstacles and the number of control points representing the path.

2.3 OPTARM III: Global path optimization

The global optimization is shown schematically in Figure 7. It computes the time optimal path between given end points without requiring an initial guess.[9] The optimization consists of a branch and bound search for the most promising initial guesses for the local optimization.

The branch and bound search is guided by a series of lower bound estimates on the optimal motion time between the given end points. The first step generates a set of K best paths, using the Dryfus search over a grid representation of the workspace. The number of paths, K, is determined from grid resolution, the desired accuracy of the solution, and a specified upper bound on the optimal motion time between the given points. These paths are then smoothed by B splines, and further pruned by the branch and bound search to reduce the initial set to a small set of best paths, each representing a neighborhood of a local minimum. These paths are then submitted as initial guesses to the local optimization (OPTARM II) to yield the global optimal solution.

This method is guaranteed to find the global optimal path under the assumptions that the optimal path is smooth and that the region of convergence around the optimal path is sufficiently large compared to the grid size. It is also assumed that small deviations in path geometry yield small deviations in the motion time along the path, so as to guarantee that the best path in the tessellated space lies in a close region to the global optimal path. Experience shows this assumption to be valid for the class of manipulators conisdered here. Solutions that do not satisfy this assumption are generally undesirable because of their high sensitivity to such factors as model errors and disturbances.

The reduction of the search to the position space drastically reduces computation time because of the typical exponential complexity of grid search methods. The use of a local optimization in conjunction with the grid search further enhances the efficiency of this approach by relaxing the demands on grid resolution.

Figure 8 shows an optimal path of a six DOF manipulator, operating in a three dimensional space. For the case shown, computation time was of the order of 30 minutes for a $10 \times 10 \times 10$ grid.

2.4 ACCMAP: Acceleration map

The acceleration map represents the acceleration capabilities of the end-effector in the workspace at zero speeds. It consists of the acceleration parallelograms, computed from the linear mapping between the actuator torques T and the joint accelerations $\ddot{q}$, assuming[10] $\dot{q} = 0$

$$\ddot{q} = M^{-1}(T - G) \tag{17}$$

If the actuator torques of a two link manipulator are bounded by constant bounds (6), then at any point in the manipulator joint space, the transformation (17) maps a rectangle, representing the region of the admissible torques, T, to a parallelogram, representing the feasible acceleration field around that point, as shown schematically in Figure 9. The acceleration parallelogram can be obtained also in work space coordinates by premultiplying the right hand side of (17) by the Jacobian matrix.[10]

Figure 10 shows the acceleration maps of a five-link manipulator at two configurations in the vertical plane. From these maps, it is evident that for the given system, the "open" configuration is too weak to sustain its own weight at some points in the workspace, as the centers of the acceleration parallelograms (represented by small crosses) are outside of the parallelograms. The "cross" configuration is clearly a "stronger" configuration at

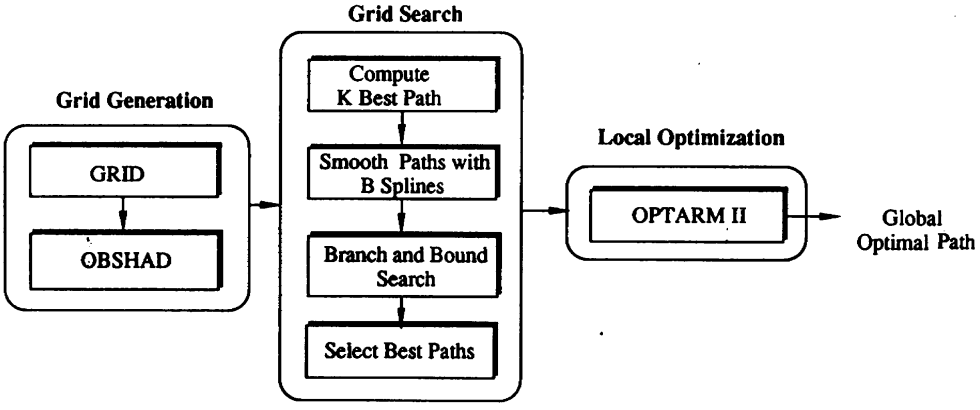


Fig. 7. Block diagram of OPTARM III.

https://doi.org/10.1017/S0263574797000052 Published online by Cambridge University Press

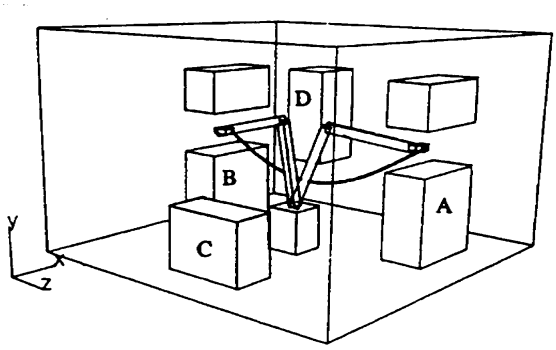


Fig. 8. An optimal path computed by OPTARM III.

most points in the workspace. This example demonsrates the use of the acceleration map for workcell design by suggesting preferred regions and directions for locating workstations in the workspace.

2.5 ACCLINES: Acceleration lines

The acceleration lines are geometric representations of the manipulator acceleration capabilities.[10] They provide a graphical tool for interactive generation of near-time optimal paths.

The acceleration lines are defined as continuous lines, generated tangent to the directions of maximum acceleration from a given point. A typical acceleration line in joint space is defined by its tangent vector, q_s (ignoring gravity for simplicity):

$$q_s = M^{-1}T^* \tag{18}$$

where T^* is some combination of the actuator extremes, corresponding to the vertices of the actuator torque region. For a two dimensional system we obtain four acceleration lines associated with the four vertices of the acceleration parallelogram. Generally, the number of acceleration lines departing from a given point is $2n$, where n is the number of DOF. The shape and orientation of the acceleration parallelogram depends on the inertia matrix M. Since M is position dependent, the acceleration lines in general are not straight lines. They are typical of the mechanical design, representing directions that combine high actuator capabilities with low effective inertia. We may view the acceleration lines as representing the ''natural'' motions for the system.

Time-optimal control is generally bang-bang, except along singular arcs.[13] Time-optimal paths are therefore tangent to the acceleration lines near the end points.[10] Figure 11 shows an example of a time-optimal path and acceleration lines in workspace coordinates for a two link

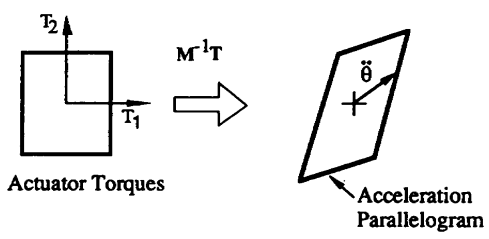


Fig. 9. Mapping actuator torques to the acceleration parallelogram.

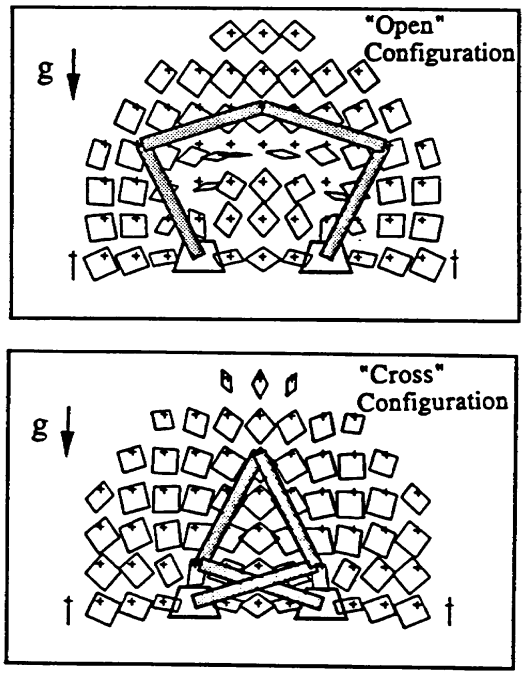


Fig. 10. The acceleration maps of a five-bar manipulator in two configurations.

manipulator. The same path is shown in the joint space in Figure 12. The optimal path, obtained independently by OPTARM II, follows closely the acceleration lines near the end points in both workspace and joint space coordinates. The high correlation between the acceleration lines and the path shown verifies the optimality of this solution. This correlation is independent of the locations of the end points and of manipulator parameters. Figure 14 shows the acceleration lines and the time-optimal path for a three DOF manipulator, with eight acceleration lines emanating from each end point.

Figure 13 shows the acceleration and configuration space obstacles in the joint space for a two link manipulator. This display clearly shows that for the case shown, the optimal path does not intersect the obstacle. Without the display of the acceleration lines, it would have been necessary to actually compute the optimal path, which is time consuming, before arriving at this conclusion.

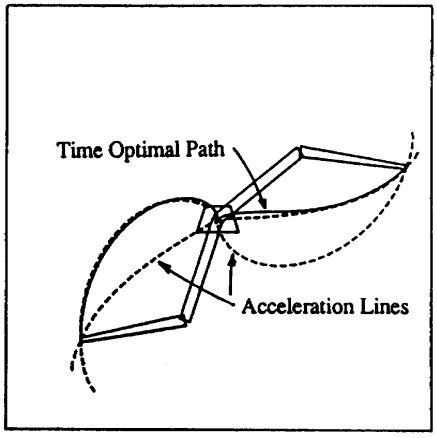


Fig. 11. Time-optimal path and acceleration lines in workspace coordinates.

https://doi.org/10.1017/S0263574797000052 Published online by Cambridge University Press

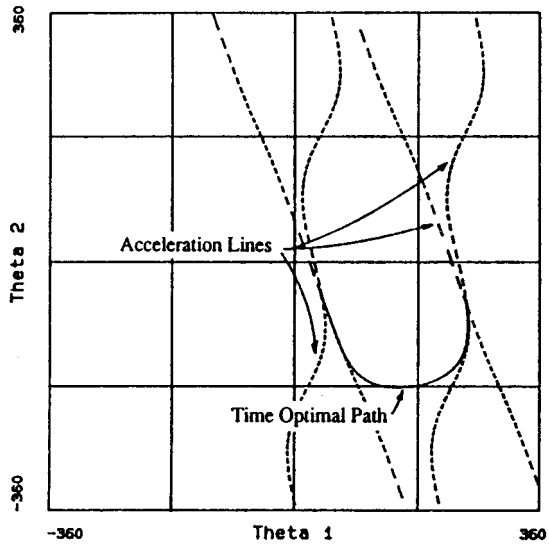


Fig. 12. Time optimal path and acceleration lines in joint space coordinates.

The acceleration lines can be used to help users select interactively near optimal paths simply by connecting smooth curves tangent to the acceleration lines near the end points. Experience shows that, guided by the acceleration lines, it is possible to approach the optimal path to within 5% or even 3% in only a few iterations on path geometry. Being in the vicinity of a local optimum, these paths make very good initial guesses for the local path optimization. The acceleration lines are also found to be a useful tool for verifying solutions obtained by other path optimization methods.

2.6 OBSHAD: Obstacle shadows

Interactive motion planning of obstacle-free paths and the design of work-cell layouts are greatly simplified with the display of obstacle shadows. An obstacle shadow is the region around an obstacle which is not accessible by the end-effector due to collision between any of the manipulator links and an obstacle. The obstacle shadow is a generalization of the configuration-space obstacle which is limited to the joint spaces.[1] The advantage of workspace representation of obstacle shadows is the intuitive insights it provides into the free space in work space coordinates—the coordinate system most intuitive to the user. It also clearly shows the correlation between obstacles and the forbidden regions they create. Figure 15 shows the obstacle shadow created by a rectangular obstacle in the workspace of a two link manipulator. In 3D, an obstacle shadow is represented by a three dimensional volume. For interactive motion planning, the obstacle shadows are displayed by two-dimensional slices in three orthogonal views, as shown in Figures 16 and 17. It is important to note that in workspace coordinates, each obstacle shadow represents the motion limits for one kinematic configuration. Multiple configurations require multiple shadows. This is not a significant problem since in many cases, only one configuration is desirable.

The obstacle shadows of planar systems can be generated by analytically computing their boundaries.[14] For more complex systems, the obstacle shadows are generated by determining the reachability of discrete points on a grid in the workspace.

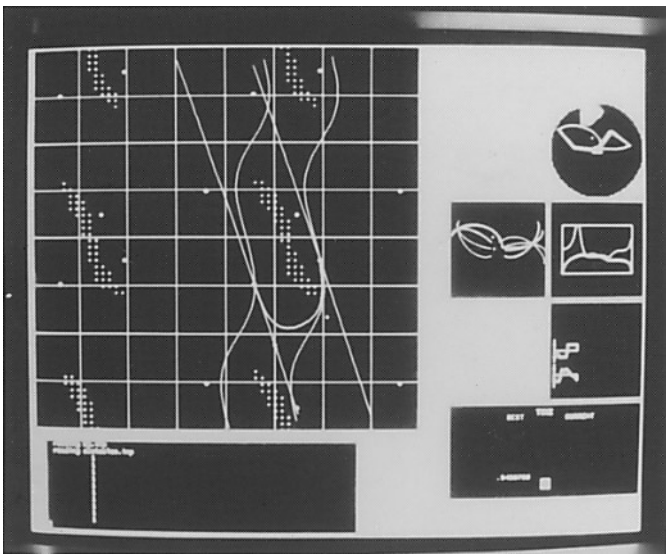


Fig. 13. A display of acceleration lines and configuration-space obstacles for a two link manipulator.

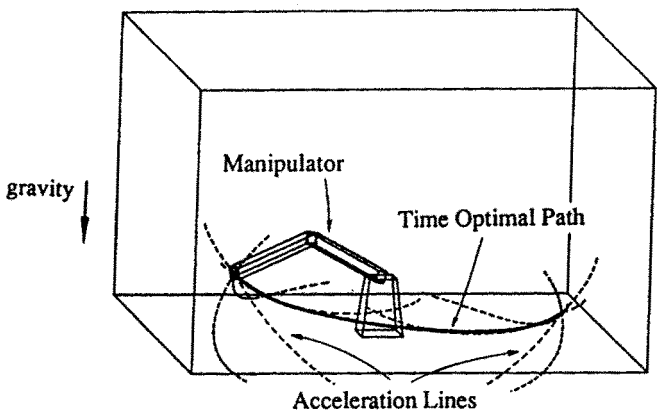


Fig. 14. A time optimal path and acceleration lines for a three DOF manipulator.

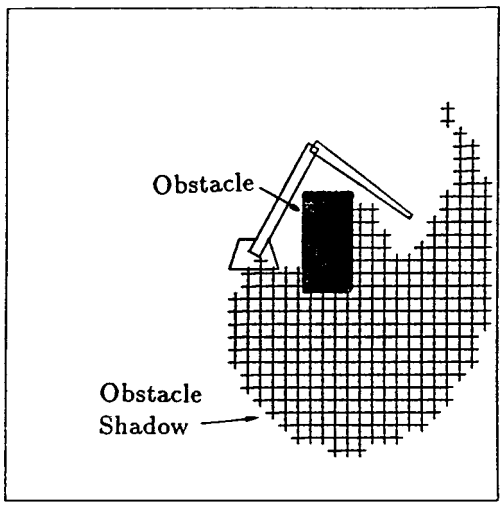


Fig. 15. A typical obstacle shadow for a two link manipulator.

https://doi.org/10.1017/S0263574797000052 Published online by Cambridge University Press

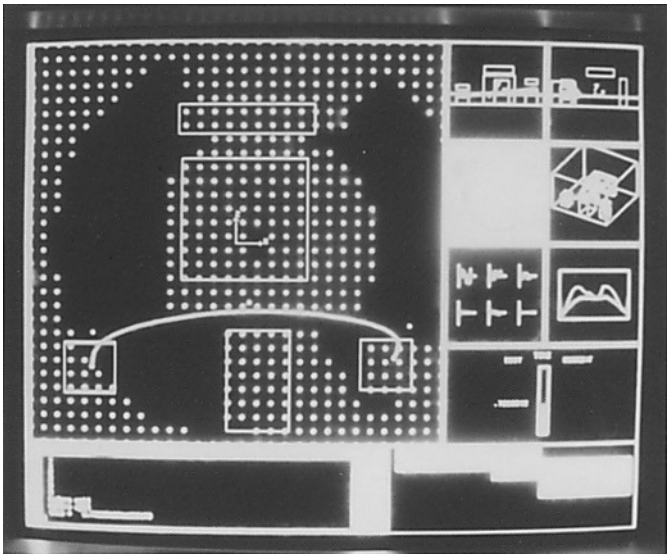


Fig. 16. A horizontal slice of an obstacle shadow of a six DOF manipulator.

3. EXAMPLE

The following example demonstrates the use of INOPT for both motion planning and work-cell design for a two link manipulator. Referring to Figure 18, the task for the manipulator is to move from point 1 to point 2, then to point 3, and finally to point 4, while avoiding obstacles A, B, C, and D, and minimizing the cycle time. All points are accessible by at least one kinematic configuration, as is evident from the obstacle shadows shown in Figure 19. However, it is not possible to find a path from point 1 to point 2 which does not cross obstacle shadows of both configurations. In fact, point 2 is isolated also from points 3 and 4. In order to accomplish the task, there are two possible alternatives, either to move point 2 to a location where it is accessible from point 1, or to clear the path to that point by moving obstacle B or C. Figure 20 shows the workcell after

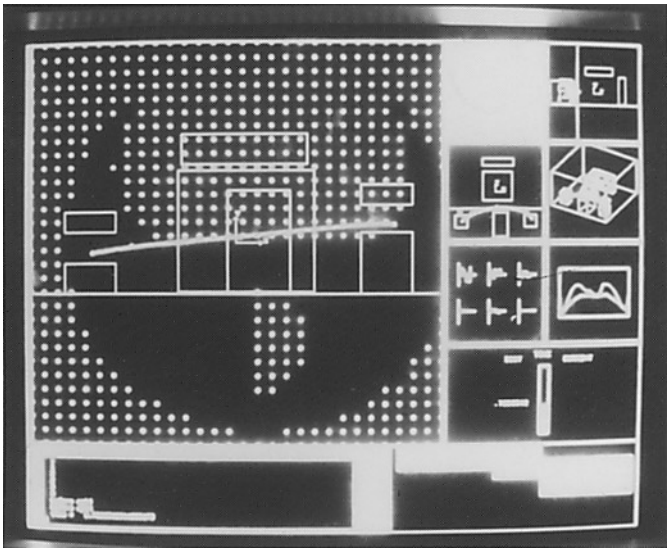


Fig. 17. A vertical slice of an obstacle shadow of a six DOF manipulator.

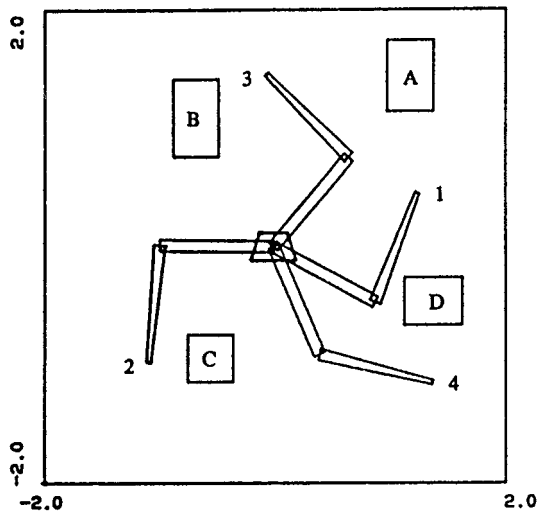


Fig. 18. Four positions to be visited by a two link manipulator.

obstacle C was moved down in the vertical direction by 0.25 m. It shows that the way to point 2 is now clear.

The next step is to obtain the time optimal paths between points 1 and 2, 2 and 3, and 3 and 4. The path between points 3 and 4 is presented here as an example for the interactive path planning. Figure 21 shows a joint space representation of points 3 and 4 and the obstacles. Each end point in work-space coordinates is represented by two points in joint space, i.e. the two solutions to the inverse kinematics for this manipulator. The points marked 31, 41 represent the kinematic configuration shown in Figure 18, while 32, 42 represent the other kinematic configuration. These points are repeated periodically in 2π.

From Figure 21 it is clear that points 32 and 42 are not reachable, which argees with the obstacle shadows in Figure 20. Point 41 is reachable from point 31 or 31*. Point 31* is a 2π repetition of point 31 in the q_2 direction. Near-optimal paths between these points were generated interactively tangent to the acceleration lines, as shown in Figure 21, with the initial traveling times of 1.280 and 0.805 seconds. Path #2, which appears to be longer, results in a shorter traversal time – it follows

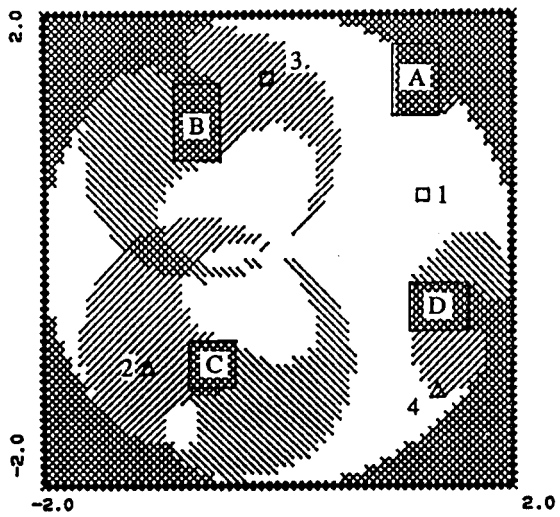


Fig. 19. Obstacle shadows for the obstacles shown in Figure 18.

https://doi.org/10.1017/S0263574797000052 Published online by Cambridge University Press

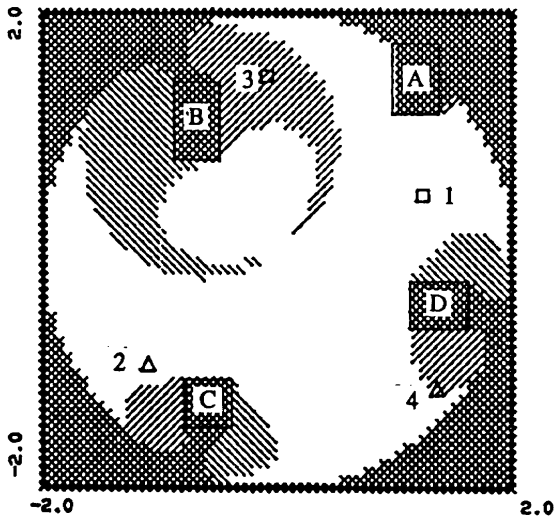


Fig. 20. Obstacle shadows after obstacle C was moved vertically by 0.25 m.

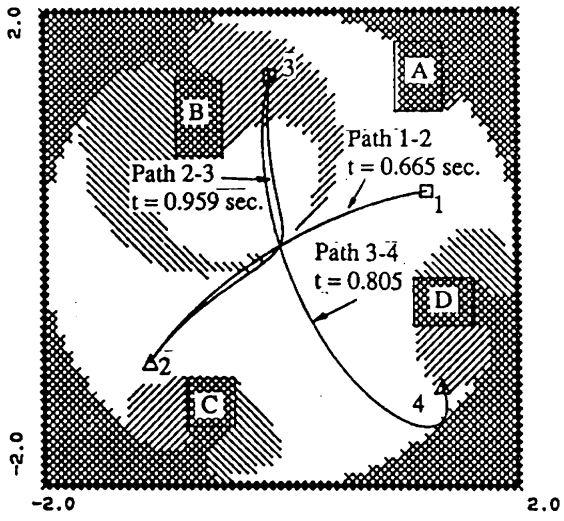


Fig. 22. Interactively obtained near-time optimal paths.

almost entirely an acceleration line. The optimization, OPTRAM II, reduced the traveling times along these paths to 1.258 and 0.764 s, respectively, with only slight changes in their geometric shapes. Clearly, Path #2 is the best selection for the task. The global optimal path, the path from point 31 to point 42° with the time of 0.682 s, is not reachable in this case.

Interactively selecting the other paths for the task, all shown in Figure 22, resulted in the cycle time of 2.429 seconds. Optimizing each path in OPTARM II reduced the cycle time to 2.368 s, only a slight improvement of 2%. The commonly used *shortest* feasible paths in joint space would result in the cycle time of 4.24 s. This demonstrates that the shortest paths, either in joint or in Cartesian spaces, are not a good choice for reducing cycle times.

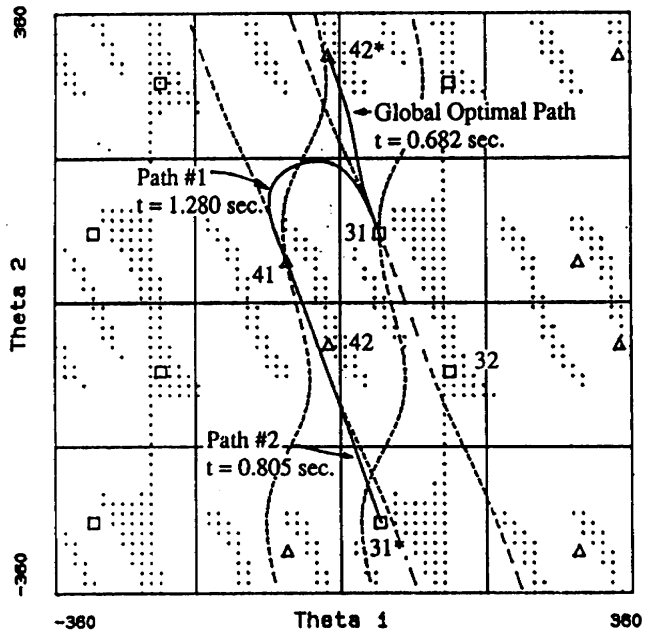


Fig. 21. Planning the path between points 3 and 4 in joint space.

4. CONCLUSIONS

The INOPT software combines graphical representations of manipulator's dynamics and obstacles with optimization modules to provide an efficient interactive and hierarchical environment for planning optimal robot motions and for work-cell layout design. The acceleration lines suggest the geometric shapes of time optimal paths. They are used to interactively plan paths for reduced motion times. Obstacle shadows help visualize the obstacle-free space for obstacle avoidance. The display of acceleration lines and obstacle shadows can be very useful for generating near-optimal and feasible paths, and for workcell design.[15] The optimization modules compute the optimal velocity profile along a given path, the local optimal path, and the global optimal path between given end points. These modules are integrated into an intuitive and easy to use system that can significantly contribute to reducing cycle times of typical robot tasks.

References

1. T. Lozano-Perez, "A Simple Motion Planning Algorithm for General Robot Manipulators" *IEEE J. of Robotics and Automation* **RA-3,** No. 3, 224–238 (June, 1987).
2. D.W. Johnson and E.G. Gilbert, "Minimum-Time Robot Path Planning in the Presence of Obstacles" *Proc. of 24th Conference on Decision and Control,* Ft. Lauderdale, FL. (1985) pp. 1748–1753.
3. Z. Shiller and S. Dubowsky, "Time Optimal Path Planning for Robotic Manipulators with Obstacles, Actuator, Gripper and Payload, Constraints" *Int. J. Robotics Research* **8,** No. 6, 3–18 (Dec., 1989).
4. M.E. Kahn and B. Roth, "The Near-Minimum-Time Control of Open-Loop Articulated Kinematic Chains" *J. Dynamic Systems, Measurement and Control,* **93,** No. 3, 164–172 (Sept., 1971).
5. J.E. Bobrow, "Optimal control of Robotic Manipulators" *Doctoral Dissertation* (Mechanics and Structures Department, School of Engineering and Applied Science, University of California, Los Angeles, 1982).
6. Z. Shiller and S. Dubowsky, "On the Optimal Control of Robotic Manipulators with Actuator and End-Effector Constraints" *Proc. IEEE Int. Conf. on Robotics and Automation,* St. Louis, MO (March, 1985) pp. 614–620.
7. Z. Shiller and H.H. Lu, "Computation of Path Constrained

https://doi.org/10.1017/S0263574797000052 Published online by Cambridge University Press

Time Optimal Motions with Dynamic Singularities'' *ASME Journal of Dynamic Systems, Measurement and Control* **114,** 34–40 (March, 1992).
8. Z. Shiller, ''On Singular Time Optimal Control Along Specified Paths'' *IEEE Transactions of Robotics and Automation* **10,** No. 4, 561–566 (August, 1994).
9. Z. Shiller and S. Dubowsky, ''On Computing the Global Time Optimal Motions of Robotic Manipulators in the Presence of Obstacles'' *IEEE Journal of Robotics and Automation* **7,** No. 6, 785–797 (Dec., 1991).
10. Z. Shiller and S. Dubowsky, ''Time Optimal Paths and Acceleration Lines of Robotic Manipulators'' *Proc. 26th IEEE Conf. on Robotics and Control,* Los Angeles, CA (December 1987) pp. 199–204.
11. J.E. Bobrow, S. Dubowsky and J.S. Gibson, ''Time-Optimal Control of Robotic Manipulators'' *Int. J. Robotics Research* **4,** No. 3, 3–17 (1985).
12. Z. Shiller, ''Time-Energy Optimal Control of Articulated Systems with Geometric Path Constraints'' *ASME Journal of Dynamic Systems, Measurement and Control* **118,** No. 1, 139–143 (March, 1996).
13. A.E. Bryson, A.E. and Y.C. Ho, *Applied Optimal Control* (Hemisphere Publishing Corp. Washington, 1975).
14. Z. Shiller and Y.R. Gwo, ''Collision-Free Path Planning of Articulated Manipulators'' *ASME Journal of Mechanical Design* **115,** No. 4, 901–908 (Dec., 1993).
15. Z. Shiller, ''An Interactive CAD-Based System for Optimal Robot Motion Planning and Work-Cell Layout Design'' *Japan-USA Symposium on flexible Automation,* Kyoto, Japan (July, 1990) pp. 447–454.

https://doi.org/10.1017/S0263574797000052 Published online by Cambridge University Press