Proceedings of the 1995 European Chinese Automation Conference, London, Sep. 1995, pp. 263-268.

An Enhanced Optimization Approach for Generating Smooth Robot Trajectories in the Presence of Obstacles

Jianwei Zhang and Alois Knoll
Faculty of Technology,
University of Bielefeld, Germany

***Abstract*—**
The basic concept of this work is the integration of path planning in environments with obstacles and motion execution under robot dynamic constraints. Instead of tracking an exactly planned geometric path, which limits not only the search space of the optimal trajectory but also the flexibility for on-line modification, a robot (mobile or manipulators) realizes its motion under the guidance of a sequence of grossly selected crucial points called subgoals.

***B*-splines are used to generate smooth trajectories through these subgoals. To optimize the motion time, first, a multi-criterion cost function is designed which integrates motion time, trajectory length and curvature. Then a hybrid optimization algorithm, with which the local minima can be explicitly detected, is developed. Experiments show that optimization results are considerably improved using the cost function and the optimization algorithm in comparison with other published approaches.**

I. Introduction

A. Problem Description

Automatic planning motion trajectories is one of the key issues for industrial robots and autonomous systems. To realize efficient robot motions, the following two factors should be considered: 1) the motion safety - the robot should move in its environments without colliding with obstacles; 2) the motion quality - the generated trajectories should be smooth and executed in minimal time (time optimal) without violating the robot dynamic constraints. Current researches about this issue can be divided in two topics: path planning and trajectory generation. Usually, a path planner computes the geometric points for the given robot and obstacle models, which represent a sequence of collision-free positions. As a subsequent component of path planning, trajectory generation considers the robot dynamic aspect and transforms the geometric data into motion parameters for each actuators. By effectively solving these two problems, the productivity of robots in industrial applications can be increased and their performance in non-industrial applications can be enhanced.

The synergy problem of path planning and trajectory generation is very complex. The pure geometric path planning is already an NP problem. Applying the concept of *configuration space* (c-space), collision-free paths should be planned in an n-dimensional c-space for a robot with n degrees of freedom (DOFs). If the control aspect of the robot trajectory is considered, the time dimension should be added to each planned path. Analytically, the dimension of the search space, which is the collection of all geometrical candidate points for the optimal trajectory and their time parameter, is infinite. The on-line control characteristic of motion control demands not only a feasible (suboptimal) but also an efficient solution.

B. State of the Art

In this work, distinctions are made between a path and a trajectory. A path of a robot includes merely the geometric information about the motion in the task space or c-space, while a trajectory is regarded as the continuous course of robot configurations over time, which can be characterized with the profiles of position, velocity and acceleration. This paper will mainly discuss the problem of trajectory generation in the presence of obstacles. Numerous approaches which have been developed for solving this problem can be briefly classified in two strategies: path-velocity decomposition and global time optimization.

The main reason to develop the strategy of path-velocity decomposition is to reduce the complexity of the coupling of path planning and time optimization by separating them in two independent steps. An important approach for time optimization of a geometric specified path was developed by Pfeiffer and Jahnanni [6] and Bobrow [1] in parallel. One distinctive advantage of this approach is that the optimization is transformed into a one-dimensional search in the phase space. However, the constraints of the permissible maximal velocity and jerk (derivative of acceleration) are not considered during the optimization because of the applied Bang-Bang control principle.

Using approaches of global time optimization, a trajectory is computed which goes through all the pre-planned data points and it is then optimized to achieve the minimal travel time. Lin et al. employed cubic splines [4], Wang [9] used B-splines to interpolate data points. In these applications, collision-avoidance was not considered and the problem of local minima, which frequently occur in the optimization process, was not explicitly treated. Shiller and Dubowsky [7], Paden et. al. [5] discussed the collision-avoidance with the time optimization. Candidate points are first selected from a grid or a quad-tree representation. Spline curves interpolate these points and they are further optimized. However, the computation cost of both approaches is very high due to the large number of data points.

C. Paper Organization

In section II, a new concept for integrating path planning and trajectory generation will be introduced. Section III will present a method for generating smooth trajectories through subgoals. Section IV will describe a hybrid algo-

rithm for optimizing trajectories to achieve minimal motion time and to avoid local collisions. Numerical examples in section V will show the advantages of this enhanced approach by demonstrating the reduction of motion time and its additional functions. Finally, section VI will give some conclusions.

II. A Concept for Integrating Path Planning and Trajectory Generation

A. Subgoal-Based Motion Control Scheme

The integrated consideration of path planning and trajectory execution distinguishes our concept of motion planning from the other approaches, see also [11]. Robot motions are planned in such a hierarchical way, that information about the motion is refined step by step, first from the connectivity features of the free space, then over the locations of representative subgoals, finally to the control parameters of each actuators. The central concept of this control scheme is the subgoal. A sequence of subgoals serves also as the common data between the path planner and the trajectory generator.

Firstly, the free space in a c-space is characterized by a finite number of free-blocks. Secondly, crucial points between the start and goal points are selected as subgoals for guiding the gross motion direction. Thirdly, trajectory generator considers the dynamic constraints of the robot, plans smooth trajectories along the subgoals and utilizes its motion capability as fully as possible.

The idea of generating subgoals is to use them for globally guiding the robot motion and still to leave some freedom for the trajectory generation to form a time optimal motion curve as well as to react on uncertainties.

B. Planning Subgoals

In many applications, the robot's environment can be at least partially modelled. The best way to make use of this environment model is to generate a plan. A robot needs a plan in order to have a global, long-term forecast of its motion course. However, the following investigation shows that it is difficult and also unreasonable to generate exact and detailed plans at the geometrical level.

A simple but important problem about robot motion along a geometric path is: "how detailed a geometric planned path should be so that the robot can fully utilize its dynamic capability and achieve the minimal motion time along this path?" A study on selecting a geometric path was carried out by Shin and McKay [8]. For manipulators, the following conclusions were drawn:

- The lower limit of the motion time increases monotonically with the length of the geometric path;
- Path segments with high curvature slow down the motion.

Based on these two conclusions, it can be proved that given the start and the goal position, if the robot must go through one more subgoal then the motion time will be longer. Therefore, a guideline for subgoal planning is as follows:

> "To achieve the time optimal motion, the number of subgoals which are used for guiding the collision-free motion should be minimal."

Besides the number of subgoals, estimations about the path length and curvature are also used as criterion for selecting subgoals. To fulfill the demands of collision avoidance with known obstacles, in our current work, subgoals are so generated that the straight connection line of each pair of neigbouring subgoals is collision-free. In [10], an approach was devised to generate subgoals in a c-space for robot manipulators by combining topological decomposition and the "rubber-band pulling" method. In [11], a tangent-graph method was employed for planning robust subgoals of minimum length for mobile robots.

C. Generating Smooth Trajectories through Subgoals

The trajectory of DOF i is regarded as a parameterized function $q^i(t)$ with the values in its permissible motion interval. The trajectory of a robot with n DOFs is then represented with the following vector:

$$\mathbf{q}(t) = [q^1(t), q^2(t), \ldots, q^n(t)]^T.$$

A trajectory is called C^k-continuous, if the k-th derivative of its position profile exists and is continuous. In this work, a trajectory is called *smooth* if it is at least C^2-continuous. In this work, B-splines are employed for generating smooth robot trajectories through all the subgoals with piecewise polynomial functions.

Fig. 1 shows the procedure of trajectory generation. Firstly, the parameters which are interpreted as the motion duration between neighbouring subgoals are initialized. Then the total motion time is optimized through adjusting these parameters with non-linear programming methods. After each optimization step, the shape of the interpolation curve will probably be changed with the parameters. Therefore, a test is performed if new collisions occur in this process. If the optimized trajectory is already collision-free, then the trajectory generation can be finished. Otherwise new subgoals will be generated and inserted at the positions where collisions occur. With the new subgoal sequence, the initialization and optimzation must be repeated until the trajectory is collision-free. Since subgoals are always selected appropriately in the free-space, the whole algorithm is guaranteed to converge.

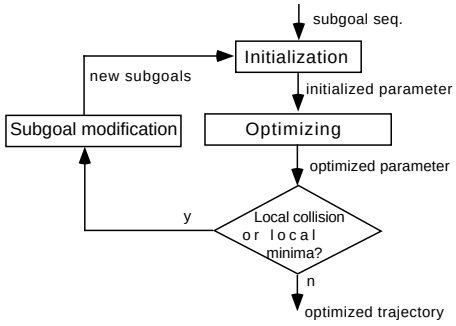

Fig. 1. Strategy of trajectory optimization

III. Subgoal Interpolation

Since distances between subgoals are usually different, *Non-Uniform-B-Splines* (in the following called B-splines for briefness) are the most suitable model for the interpolation task, [2]. Based on a set of basis functions, control points (or *de Boor* points) are computed from the data points to be interpolated. The basis functions and these control points specify then a unique smooth curve.

A. Problem Representation

Assume that m data points $\mathbf{q}_0, \mathbf{q}_1, \ldots, \mathbf{q}_m$ should be interpolated. The following well-ordered parameter values are defined for the basis functions of the B-Splines of order k:

$$t_0 < t_1 < \ldots < t_m < \ldots < t_{m+k}$$

These parameter values are called as knots. If the normalized basis functions are noted as $N_{j,k}$ (see [2] for their computation). A B-spline curve of order k can be constructed through blending a set of control points with the basis functions:

$$\mathbf{q}(t) = \sum_{j=0}^{m} \mathbf{v}_j N_{j,k}(t), \quad t \in [t_{k-1}, t_{m+1}] \tag{1}$$

(1) is a piecewise defined polynomial of degree $k-1$, it describes the curve course in each coordinate of the trajectory in the n-dimensional c-space over parameter t.

The control points $\mathbf{v}_j$ are normally not identical with the data points for the interpolation, but can be determined by solving the following equation system (2):

$$\mathbf{q}_j(\tau_j) = \sum_{j=0}^{m} \mathbf{v}_j N_{j,k}(\tau_j), \quad j = 0, 1, \ldots, m. \tag{2}$$

B. Parameter Initialization

The $m+1$ subgoals $\mathbf{q}_0, \mathbf{q}_1, \ldots, \mathbf{q}_m$ must be parameterized with $\tau_0, \tau_1, \ldots, \tau_m$. Given a subgoal sequence $\mathbf{q}_0, \mathbf{q}_1, \ldots, \mathbf{q}_m$, the initial values τ_j can be determined based on different parametrization strategies such as equidistant, chordal or centripedal parametrization, [2]. Since the subgoals are generated according to the topology and geometry of components of the free-space and possess variant distances, a reasonable initialization of τ_j is the chordal parametrization.

The distances between subgoals are computed as follows:

$$\Delta_j = \begin{cases} 0 & : \quad j = 0 \\ |\mathbf{q}_{j-1}\mathbf{q}_j| & : \quad j = 1, \ldots, m \end{cases}$$

The parameter values τ_j of subgoal $\mathbf{q}_j$ can be initialized as the estimated motion duration from $\mathbf{q}_0$ to $\mathbf{q}_j$:

$$\tau_j = \frac{\sum_{k=0}^{j} \Delta_k}{\sum_{j=0}^{m} \Delta_j}, \quad j = 0, \ldots, m, \tag{3}$$

The vector $\tau = [\tau_0, \tau_1, \ldots, \tau_m]^T$ includes all the parameters for the optimization.

C. End Condition of Interpolation

Besides the geometric constraints, several end conditions of the trajectory should be considered for solving the control points. The robot moves from the start point $\mathbf{q}_0$ through intermediate subgoals to subgoal $\mathbf{q}_m$. Important end conditions are the demands on the velocity and acceleration at $\mathbf{q}_0$ and $\mathbf{q}_m$. Typically, the robot starts at $\mathbf{q}_0$ from zero velocity and moves to $\mathbf{q}_m$ to zero velocity again. Therefore, the following end conditions of the interpolation are generated:

$$\begin{aligned} \dot{\mathbf{q}}(\tau_0) = \ddot{\mathbf{q}}(\tau_0) = \mathbf{0} \\ \dot{\mathbf{q}}(\tau_m) = \ddot{\mathbf{q}}(\tau_m) = \mathbf{0} \end{aligned} \tag{4}$$

The consideration of these four end conditions in the equation system (2) then leads to a quadruplication of the first and the last parameters:

$$\begin{aligned} & (\tau_0, \tau_0, \tau_0, \tau_1, \tau_2, \ldots, \tau_m, \tau_m, \tau_m) \\ & \text{after an index transformation} \\ = \; & (\tau_0, \tau_1, \tau_2, \tau_3, \tau_4, \ldots, \tau_{m+2}, \tau_{m+3}, \tau_{m+4}) \\ = \; & (\tau_0, \tau_1, \ldots, \tau_{m+c}) \end{aligned}$$

where

$$\begin{aligned} & \tau_0 = \tau_1 = \ldots = \tau_{\frac{c}{2}}, \\ & \tau_{m+\frac{c}{2}} = \tau_{m+\frac{c}{2}+1} = \ldots = \tau_{m+c}, \\ & m+1: \text{ number of the subgoals}, \\ & k: \text{ order of the } B\text{-splines}, \\ & c: \text{ number of the end conditions}. \end{aligned}$$

The equation system (2) has one unique solution if it is valid:

$$t_j \leq \tau_j < t_{j+k} \quad \text{for all} \quad j = 0, \ldots, m+c-k. \tag{5}$$

(5) is called Schoeneberg-Whitney condition. To fulfill these conditions, the knots can be determined as follows:

$$t_{j+k} = \frac{1}{k-1} \sum_{l=j+1}^{j+k-1} \tau_l \quad j = 0, 1, \ldots, m+c-k.$$

Here the rest of end knots are repeated k-time:

$$\begin{aligned} t_0 = t_1 = \ldots = t_{k-1} = \tau_0 \\ t_{m+c+1} = t_{m+c+2} = \ldots = t_{m+c+k} = \tau_{m+c} \end{aligned}$$

IV. A Hybrid Optimization Algorithm

The number of the parameters to be optimized τ_j, $(j = 0, \ldots, m)$ can be quite large. Therefore, the following measures are advised to accelerate the optimization process and to overcome the problem of local minima arisen in the optimization.

A. Time Scaling

The boundary of the minimal motion time of a sub-trajectory $\mathbf{q}_j^i(t)$ is determined by the robot motion capacity represented by the dynamic parameters of all actuators of each DOF. For DOF i, these constraints can be represented as follows:

$$|\dot{q}_j^i(t)| \leq \dot{q}_{max}^i \tag{6}$$

$$|\ddot{q}_j^i(t)| \leq \ddot{q}_{max}^i \tag{7}$$

$$|u_j^i(t)| \leq u_{max}^i \tag{8}$$

where i $(i = 1, \ldots, n)$ is the DOF index, and j $(j = 1, \ldots, m)$ represents the index of the sub-trajectory, u^i is the torque of DOF i.

One heuristic of the time optimal motion is to fully utilize the dynamic capacity during the execution of the trajectory $\mathbf{q}(t)$. To realize this heuristic, the sub-trajectory $\mathbf{q}_j(t)$ in each parameter interval $[\tau_{j-1}, \tau_j]$ $(j = 0, \ldots, m)$ is adjusted through scaling of time τ to the upper limit of the boundary curve.

Noted the old time parameter as t_{alt}, the adjusted time parameter after the scaling with a factor λ is then:

$$\tau = \lambda \cdot \tau_{alt}$$

The derivatives of $\mathbf{q}(t)$ are:

$$\dot{\mathbf{q}} = \lambda \cdot \dot{\mathbf{q}}_{alt}$$

$$\ddot{\mathbf{q}} = \lambda^2 \cdot \ddot{\mathbf{q}}_{alt}$$

The quotients of the values from the trajectory and the maximal permissible values

$$\lambda_{j,v} = \max_{i=1,\ldots,n} \left[\frac{|\dot{q}_j^i|}{\dot{q}_{max}^i} \right] \tag{9}$$

$$\lambda_{j,a} = \max_{i=1,\ldots,n} \left[\frac{|\ddot{q}_j^i|}{\ddot{q}_{max}^i} \right] \tag{10}$$

$$\lambda_{j,u} = \max_{i=1,\ldots,n} \left[\frac{|u_j^i(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}})|}{u_{max}^i} \right] \tag{11}$$

can be used as the measurements for the full utilization of the dynamic capacity. The maximal value:

$$\lambda_j = \max(\lambda_{j,v}, \sqrt{\lambda_{j,a}}, \sqrt{\lambda_{j,u}}) \tag{12}$$

describes the overall capacity utilization and

$$R_j = 1 - \lambda_j \tag{13}$$

the remained local dynamic reserve in $[\tau_{j-1}, \tau_j]$, Fig. 2.

From the consideration that the dynamic capability should be fully utilized at each position, the following modified total motion time is devised:

$$T = \sum_{j=\frac{c}{2}+1}^{m+\frac{c}{2}} (\tau_{j+1} - \tau_j)(1 + |R_j|) \tag{14}$$

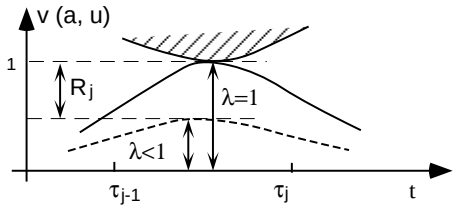

Fig. 2. Full utilization of dynamic reserve

B. Design of the Cost Function

Analytically, the total motion time T of (14) is the final cost function of the optimization. However, experiments have shown that no satisfied optimization results can be achieved if T is applied as the single cost function. The main problem is still the local minimum. Based on the conclusions of section II.B, the trajectory length, curvature are the two most important factors for achieving the minimal motion time. It is then reasonable to integrate them in the cost function.

Given a curve in an n-dimensional c-space:

$$\mathbf{q}(t) = [q^1(t), q^2(t), \ldots, q^n(t)]^T$$

and two points $\mathbf{p}_0 = \mathbf{q}(t_s)$ and $\mathbf{p}_1 = \mathbf{q}(t_z)$, then the trajectory length L between $\mathbf{p}_0$ and $\mathbf{p}_1$ is the integral:

$$L = \int_{t_s}^{t_z} |\dot{\mathbf{q}}(t)| \, dt \tag{15}$$

Given the parameter t, the first derivation $\dot{\mathbf{q}} = d\mathbf{q}(t)/dt$ and the second derivation $\ddot{\mathbf{q}} = d\dot{\mathbf{q}}(t)/dt$ of the trajectory $\mathbf{q}(t)$, the curvature $\kappa(t)$ can be computed, see [2]. After that, the *bending energy* of a smooth curve $\mathbf{q}(t)$ over the interval $t \in [0, T]$ is defined as

$$E = \int_0^T \kappa(t)^2 |\dot{\mathbf{q}}(t)| \, dt \tag{16}$$

Bending energy is invariant to translation and rotation but not to the scaling. For the purpose of time optimization, bending energy alone is not a suitable cost function, since it always strikes for an enlargement of the parameter interval.

Usually, there exist conflicts between these factors. Therefore, a compromise between these factors should be made. To consider all these three factors together, the following cost function is proposed:

$$f = T \cdot L \cdot E \tag{17}$$

Based on this cost function, the simultaneous optimization of the trajectory length L and the bending energy E also contributes to the minimization of motion time.

C. Detecting Local Minima and Eliminating Local Collisions

A module is employed to explicitly detect the possible local minima and local collisions. The detection of local

minima is based on the following principle: a time optimal trajectory should possess not only a short length but also a small bending energy. Therefore, the sub-trajectory between two subgoals should not depart too much from the straight line connecting them. Then a local minimum in a local area can be found out by checking if the curve length is considerably larger than the corresponding edge length, see Fig. 3.

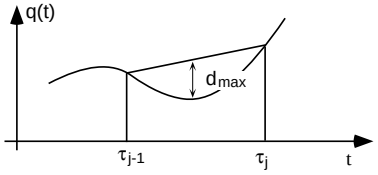

Fig. 3. Geometrical detection of a local minimum

As a measurement for the departure, the maximal distance d_{max} between the edge and the curve is introduced:

$$d_{max} = \max_{\tau_{j-1}<t<\tau_j} |q(t) - k(t)| \qquad (18)$$

where $q(t)$ represents the curve value and $k(t)$ edge value at t.

Similarly, local collisions can be detected by checking if the generated trajectory curve intersects with the c-obstacles. If such a local collision occurs, the intersection points are computed.

If local minima or local collisions are detected, a geometric module for the treatment will be applied. By inserting additional subgoal points using the "rubber-band pulling" method, the trajectory curve can be forced to come out from a local minimum (which works like curve smoothing), or the local collisions can be eliminated (which works like local repairing).

D. Optimization Using Nonlinear Programming

There exist numerous approaches of nonlinear optimization, see [3]. For optimizing a B-spline trajectory, a hybrid algorithm is developed which consists of the gradient method, the polyhedron search and time scaling. The gradient method and polyhedron search compensate each other. Additionally, time scaling is executed each time before evaluating the cost function. From the computation (12), a common scaling factor λ for the overall trajectory $\mathbf{q}(t)$ can be represented as:

$$\lambda = \max_{j=1,\ldots,m} \lambda_j$$

The time parameter vector τ is scaled until:

$$\lambda = 1.$$

V. Numerical Examples

Various examples in two- to six-dimensional c-spaces have been tested with the hybrid optimization algorithm and the integrated cost function described above. The examples shown in [4] and [9], in which no collision avoidance was considered, have been tested again. The comparison results show that the improvement of the optimization accounts to 20% to 60%.

A. Example 1 – Optimization of a Trajectory for a PUMA-600

A manipulator of type PUMA 600 with six DOFs is considered, its dynamic constraints are listed in Tab. 1, see [9].

Joint	Velocity (degree/s)	Acceleration (degree/s^2)	Jerk (degree/s^3)
1	110.0	45.0	60.0
2	95.0	40.0	60.0
3	100.0	75.0	55.0
4	150.0	70.0	70.0
5	130.0	90.0	75.0
6	110.0	80.0	70.0

Table 1. Dynamic constraints of a PUMA-600 Manipulator

Ten subgoal points, which are directly selected in a c-space, are shown in Tab. 2.

No.	Joint 1 (degree)	Joint 2 (degree)	Joint 3 (degree)	Joint 4 (degree)	Joint 5 (degree)	Joint 6 (degree)
1	15.0	10.0	50.0	15.0	10.0	6.0
2	30.0	25.0	70.0	20.0	30.0	20.0
3	50.0	30.0	150.0	40.0	10.0	40.0
4	90.0	15.0	200.0	80.0	-40.0	80.0
5	130.0	-20.0	120.0	80.0	-60.0	70.0
6	90.0	-55.0	35.0	40.0	10.0	10.0
7	45.0	-70.0	-10.0	-60.0	50.0	-10.0
8	-10.0	-20.0	50.0	-100.0	-40.0	15.0
9	-30.0	0.0	60.0	-60.0	-20.0	30.0
10	-50.0	10.0	50.0	-30.0	10.0	20.0

Table 2. Subgoal points to be interpolated

The B-splines model of order 5 is employed for the interpolation. Fig. 4 depicts the optimization results with the trajectory profiles for joint 1 of the six joints. In each single figure, the horizontal axis represents the motion time and the vertical axis the values of the following variables:

- P_i: position profile of joint i;
- V_i: velocity profile of joint i;
- A_i: acceleration profile of joint i;
- J_i: jerk profile of joint i;

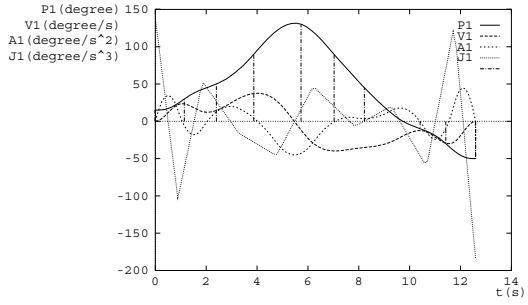

Fig. 4. Examples of optimizing a trajectory of a six joint PUMA robot - joint 1

A vertical broken segment in Fig. 4 illustrates the value of a subgoal, its abscissa the time point at which the subgoal was achieved.

B. Example 2 – Optimization in a C-space with C-obstacles

For clarirty, the same example in section II.B is further used for the trajectory optimization. In Fig. 5, two c-obstacles, the subgoals and the optimized trajectory with B-splines of order 5 are illustrated. It is noticeable that the optimized trajectory still keeps a certain safety distance to the c-obstacles because the original obstacles in the task space are assumed to possess uncertainties. It can be seen from this figure that a time optimized trajectory is not the shortest but achieves a compromise between the arc length and the curvature.

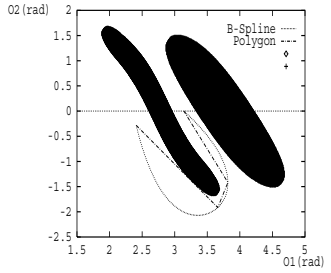

Fig. 5. Optimized trajectory through the subgoals in the c-space

Fig. 6 depicts the profiles of the position (P_i), the velocity (V_i), the acceleration (A_i) and the jerk (J_i) of joint i ($i = 1, 2$).

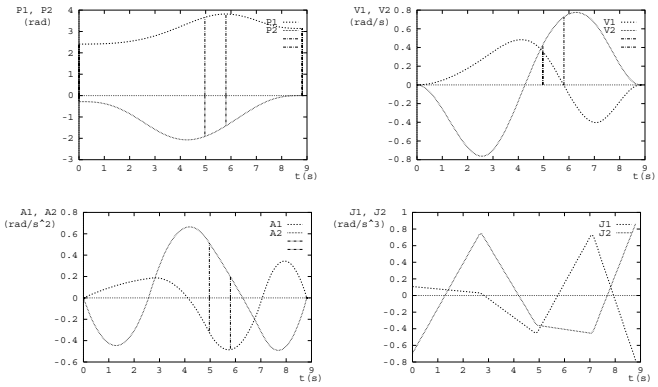

Fig. 6. Trajectory profiles of two joints passing through subgoals

VI. Conclusions

Based on this enhanced optimization approach, piecewise polynomial trajectories can be generated which pass through all given subgoals without violating the dynamic constraints. As an example, the generated trajectories with B-splines of order 5 are C^3-continuous. The important conclusions are summarized in the following:

- The interpolation formulation is valid for any order of B-splines, i.e. any degree of trajectory smoothness can be achieved according to the demands of different applications;
- A cost function designed for optimization integrates the factor of motion duration, trajectory length and bending energy. With this cost function, multiple factors which influence the motion time drive the optimization algorithm jointly towards the minimal result;
- Experiments with the hybrid optimization algorithm and the integrated cost function show that considerable improvement can be achieved in comparison with the existing methods. The computation time for these trajectories is acceptable (ranging from seconds to minutes in a SUN-Sparc workstation).
- This approach supplies additional possibilities to detect local minima as well as local collisions during the optimization process and to eliminate them by inserting new subgoal points.

References

[1] J. E. Bobrow. Optimal robot path planning using the minimum-time criterion. *IEEE Journal on Robotics and Automation*, 4:443–450, August 1988.

[2] B. K. Choi. Surface modeling for CAD/CAM. *ELSEVIER*, 1991.

[3] D. M. Himmelblau. *Applied Nonlinear Programming*. McGraw-Hill, New York, 1972.

[4] C.-S. Lin, P.-R. Chang, and J. Y. Luh. Formulation and optimization of cubic polynomial trajectories for industrial robots. *IEEE Transactions on Automatic Control*, AC-28(3):1066–1073, 1983.

[5] B. Paden, A. Mees, and M. Fisher. Path planning using a jacobian-based freespace generation algorithm. *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1732–1737, 1989.

[6] F. Pfeiffer and R. Jahnanni. A concept for manipulator trajectory planning. *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1399–1405, 1986.

[7] Z. Shiller and S. Dubowsky. On computing the global time optimal motions of robotic manipulators in the presence of obstacles. *IEEE Transactions on Robotics and Automation*, 7(6):785–797, December 1991.

[8] K. G. Shin and N. D. McKay. Selection of near-minimum time geometric paths for robot manipulators. *IEEE Transactions on Automatic Control*, AC-31(6):501–511, June 1986.

[9] K. Wang. B-splines joint trajectory planning. *Computers in Industry*, 10:113–122, 1988.

[10] J. Zhang. An integrated method for planning smooth collision-free trajectories for robot arms. *Proceedings of Third Annual Conference on AI, Simulation and Planning in High Autonomy Systems, Perth*, pages 46–53, 1992.

[11] J. Zhang and J. Raczkowsky. Robust subgoal planning and motion execution for robots in fuzzy environments. *Proceedings of Internaltional Conferece on Intelligent Robots and Systems (IROS), Munich*, 1994.