Contents lists available at ScienceDirect

Robotics and Computer–Integrated Manufacturing

journal homepage: www.elsevier.com/locate/rcim

Optimal trajectory generation algorithm for serial and parallel manipulators

Serdar Kucuk

Kocaeli University, Technology Faculty, Department of Biomedical Engineering, Umuttepe Campus, 41380 Kocaeli, Turkey

ARTICLE INFO

Keywords:
Trajectory generation
Cubic spline
Jerk
PSO algorithm
Serial and parallel manipulators

ABSTRACT

In this paper, an **O**ptimal **T**rajectory **G**eneration **A**lgorithm (**OTGA**) is developed for generating minimum-time smooth motion trajectories for serial and parallel manipulators. OTGA is divided into two phases. The first phase encompasses derivation of minimum-time optimal trajectory using cubic spline due to its less vibration and overshoot characteristics. Although cubic splines are widely used in robotics, velocity and acceleration ripples in the first & last knots can worsen manipulator trajectory. The second phase includes changing cubic spline interpolation in the first and last knots of optimized trajectory with 7th order polynomial for having zero jerk at the beginning and end points of trajectory. Performing this modification eliminate undesired worsening in the trajectory and provide smoother start and stop of joint motions. **P**article **S**warm **O**ptimization (**PSO**) is chosen as optimization algorithm because of its easy implementation and successful optimization performance. OTGA has been tested in simulation for PUMA robot and results are compared with algorithms proposed by earlier authors. In addition, a discrete-time PID control scheme for PUMA robot is designed for comparing energy consumption of OTGA with algorithms developed by previous authors. Comparison results illustrated that OTGA is the better trajectory generation algorithm than the others.

1. Introduction

Trajectory generation is very important for robotic manipulators since it produces input to the control system of the robotic manipulators for executing the desired task with satisfactory performance [1]. In addition, generated trajectories must provide smooth kinematic motion to maintain high tracking accuracy and avoid exciting the natural modes of control system [2]. In industry, most of the robot trajectories are first constructed off-line and then end-effectors of robots are forced to track this path on-line. Off-line trajectory generation can be performed at two different approaches namely, hand level and joint level. Minimum time trajectory generation at hand level was performed by the authors of Luh and Lin [3]. In hand level approach, joint coordinates are converted into Cartesian coordinates by means of Jacobian transformation for each sampling period. Errors may occur in inverse Jacobian transformation therefore robot tracks the trajectory incorrectly. Afterwards reference input torque for each joint is calculated by means of robot dynamic equations which are highly nonlinear [4]. The major drawback of the hand level approach is that whole process is performed at one sampling period. This is always limited and decelerates the system performance [5]. In joint level approach, Cartesian coordinates are converted into joint coordinates by means of inverse kinematics. Robot manipulators are controlled at joint level which is less expensive in terms of computational complexity compared to hand level approach. In joint level approach, kinematic constraints are considered only during the trajectory generation [6]. The dynamic constraints that increase the computational effort are ignored. It is the major advantage of the joint level approach. Hence it is mostly preferred to reduce computational effort.

In joint level approach, a number of trajectory points (via points) is firstly defined in terms of reference end-effector pose in Cartesian space. Secondly these via points (called also as knots) are transformed into joint angles by using inverse kinematics. Finally these knots are interpolated using splines. Several kinds of splines such as polynomial, trigonometric [7], quantic [8,9], B-spline [10] and cubic spline [5] are proposed for trajectory generation in literature. Among these splines, cubic spline is mostly used for interpolating the robot trajectories since it provides continuity in velocity and acceleration at every knot. It should be noticed that cubic spline curves do not provide continuity in jerk. Although cubic spline curves does not provide continuity in jerk, they are only the third-degree spline that can provide jerk limitation [11]. Cubic splines can also produce lowest possible jerk peak [8]. This feature is especially important for reducing vibration. Thus a smooth curve trajectories can be obtained during motion. Minimizing jerk decreases joint position errors, causes less vibration, limits excessive wear on the robot and prevents large oscillations which can occur employing higher order polynomials [11–14].

Although cubic splines are commonly used in robotics due to its several advantageous mentioned above they have an important drawback. Joint velocity and acceleration ripples which take place in the

E-mail address: skucuk@kocaeli.edu.tr.

http://dx.doi.org/10.1016/j.rcim.2017.04.006
Received 31 August 2016; Received in revised form 13 April 2017; Accepted 13 April 2017

first and last knots can deteriorate the manipulator trajectory [15]. Several studies have been performed to generate smooth and time-optimal trajectories by overcoming this problem in the literature. Lin et al. [5] formulated and optimized cubic polynomial joint trajectories for using polyhedron search method. Piazzi and Visioli [6] studied optimization of global minimum-jerk trajectory planning of robot manipulators using interval analysis. Tandu and Bazaz [16] developed three-cubic methods to generate a joint trajectory by interpolating intermediate positions and velocities based on a combination of cubic splines. They used analytical optimization approach. Chettibi et al. [17] presented minimum cost trajectory planning algorithm for robotic manipulators using Sequential Quadratic Programming (SQP) method. Gasparetto and Zanotto [18] proposed a technique for time-jerk optimal planning of robot trajectories. They used sequential quadratic programming techniques in order to get the optimal trajectory. Kolter and Andrew [15] developed a method for optimizing task-space cubic spline trajectories using convex optimization techniques. Demeulenaere et al. [19] developed a general framework to synthesize optimal polynomial splines for rigid motion systems using convex programming framework. Aribowo and Terashima [11] performed a study about cubic spline trajectory planning and vibration suppression of semiconductor wafer transfer robot arm. They used SQP method for solving the constrained nonlinear trajectory planning optimization problem. The authors mentioned above have used classic numerical optimization algorithms based on successive linearization using the first and the second derivatives of objective functions in trajectory planning of robotic manipulator. The disadvantages of these derivative-based optimization algorithms arise from sensitivity to problem formulation and algorithm selection. They usually converge to a local minimum [20]. The PSO algorithm used also in this study have certain advantages over classical optimization techniques and other evolutionary algorithms: (i) It finds optimal or near optimal solutions to nonlinear and discontinuous problems at higher dimensions within the shorter computation time, (ii) it has memory that makes the knowledge of the better solution obtained from last iterations to save all the particles, (iii). The solution does not depend on the initial population. Therefore PSO algorithm is considered an important part of OTGA proposed in this study. Trajectory generation is performed into phases.

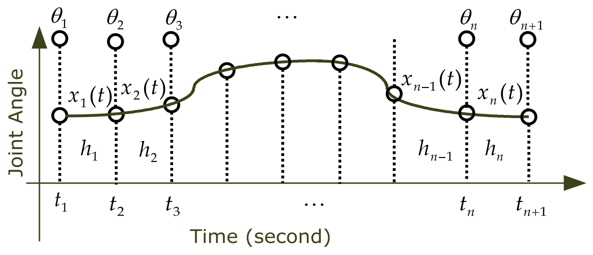

Fig. 1. Cubic spline trajectory with several piecewise cubic segments.

2. Formulation of cubic spline joint trajectory

In robotics, there are two types of motion used in general namely, pick & place motion and path-constraint motion. In pick and place motion, start and end-points are important. Robot manipulators perform a task freely between pick and place locations. However in constraint motion, robot manipulators follow a certain path that must be defined in advance [21]. Path constraint motion is of vital importance such as in welding, cutting, surgery and machining applications where continuous path motions are required.

Polynomial splines are often applied to perform path constraint motion in general. Among polynomial splines, cubic splines are often preferred since they provide continuous velocity and acceleration with lowest degree [5]. Cubic spline interpolation for robot manipulators uses several kinematically feasible Cartesian knots between start and end-points. These knots defined in Cartesian task space are then converted into joint space by using inverse kinematics. Subsequently each joint trajectory is planned as a combination of numerous piecewise cubic segments $(x_1(t), x_2(t),\ldots,x_{n-1}(t), x_n(t))$ that connects predefined knots as in Fig. 1. A joint trajectory with n piecewise cubic segments possess n+1 predetermined joint angle values $(\theta_1, \theta_2,\ldots,\theta_n, \theta_{n+1})$ [22].

The following system of linear equations can be derived by performing a sequence of mathematical operations as in [22]. The joint acceleration $(\ddot{\theta}_2, \ddot{\theta}_3,\ldots,\ddot{\theta}_{n-1}, \ddot{\theta}_n)$ at each knot can be found by solving following system of linear equations where h_i represents the time intervals between $t_i$and t_{i+1}, $(i = 1,2,\ \ldots,\ n + 1)$.

$$\begin{bmatrix} h_1^2+3h_1h_2+2h_2^2 & h_2^2 & 0 & 0 & 0 & 0 & 0 \\ 3(h_1+h_2) & 2h_3+3h_2 & h_3 & 0 & 0 & 0 & 0 \\ & h_3 & 2(h_3+h_4) & h_4 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & h_{n-3} & 2(h_{n-3}+h_{n-2}) & h_{n-2} & 0 \\ 0 & 0 & 0 & 0 & -h_{n-2} & -3h_{n-1}-2h_{n-2} & -3(h_n+h_{n-1}) \\ 0 & 0 & 0 & 0 & 0 & -h_{n-1}^2 & -(h_n^2+2h_{n-1}^2+3h_{n-1}h_n) \end{bmatrix} \begin{bmatrix} \ddot{\theta}_2 \\ \ddot{\theta}_3 \\ \ddot{\theta}_4 \\ \vdots \\ \ddot{\theta}_{n-2} \\ \ddot{\theta}_{n-1} \\ \ddot{\theta}_n \end{bmatrix}$$
$$= 6\begin{bmatrix} \theta_3-\theta_1 & \frac{\theta_4-\theta_3}{h_3} & \frac{\theta_5-\theta_4}{h_4}-\frac{\theta_4-\theta_3}{h_3} & \cdots & \frac{\theta_{n-1}-\theta_{n-2}}{h_{n-2}}-\frac{\theta_{n-2}-\theta_{n-3}}{h_{n-3}} & \frac{\theta_{n-1}-\theta_{n-2}}{h_{n-2}} & \theta_{n+1}-\theta_{n-1} \end{bmatrix}^T \tag{1}$$

The first phase includes derivation of minimum-time optimal trajectory using PSO algorithm. Although cubic spline optimization is useful in minimizing the joint velocity, acceleration ripples and jerk, the first and last knots have still a sharp start and stop of motion that can deteriorate manipulator trajectory. In the second phase, the cubic spline interpolation in the first and last knots of optimized trajectory are changed with the interpolation of seventh order polynomial in order to get rid of this undesired deteriorations. Performing this change makes the joints have smoother start and stop of motion. Subsequently, OTGA has been tested in simulation for PUMA robot and the results are compared with the previous algorithms. Electrical energy consumptions of OTGA and the previous algorithms are also compared and given in a table. Finally comparison results are discussed.

3. Particle swarm optimization

PSO algorithm introduced by Kennedy and Eberhart in 1995 [23] is a robust stochastic population-based technique for solving numerical optimization problems. The PSO algorithm (whose flowchart given in Fig. 2) have certain advantages over classical optimization techniques and other evolutionary algorithms: (i) it finds optimal or near optimal solutions to nonlinear and discontinuous problems at higher dimensions, (ii) it is computationally inexpensive, (iii) it has fewer parameters to adjust, (iv) it has memory that makes the knowledge of the better solution obtained from last iterations to save all the particles, (v) quickly converge the fitness function and (vi) the solution does not depend on the initial population. PSO algorithm has been successfully applied in many kinds of engineering problems [24–26]. In particle swarm optimization technique, every