Efficient Trajectory Optimization for Robot Motion Planning

Yu Zhao, Hsien-Chung Lin, and Masayoshi Tomizuka

***Abstract*— Motion planning for multi-jointed robots is challenging. Due to the inherent complexity of the problem, most existing works decompose motion planning as easier subproblems. However, because of the inconsistent performance metrics, only sub-optimal solution can be found by decomposition based approaches. This paper presents an optimal control based approach to address the path planning and trajectory planning subproblems simultaneously. Unlike similar works which either ignore robot dynamics or require long computation time, an efficient numerical method for trajectory optimization is presented in this paper for motion planning involving complicated robot dynamics. The efficiency and effectiveness of the proposed approach is shown by numerical results. Experimental results are used to show the feasibility of the presented planning algorithm.**

I. INTRODUCTION

Motion planning for robots with multi-jointed arms is challenging. Due to the complicated geometric structure and nonlinear dynamics, time-consuming computation is required to solve motion planning problem even in the simplest cases. Most existing works utilize the path-velocity decomposition approach [1], in which motion planning problem is separated into easier subproblems, i.e., path planning and trajectory planning. The path planning problem focuses on the generation of collision free geometric path in the configuration space, while the trajectory planning problem focuses on the generation of time optimal velocity profile along the geometric path. Extensive research [2], [3], [4], [5] has been conducted for each subproblem, resulting a rich collection of algorithms. However, due to the inconsistency of performance metrics between motion planning problem and the subproblems, only sub-optimal solution can be found by path-velocity decomposition based approaches. Time optimal motion planning involving collision avoidance requirement and robot dynamics is still a challenging problem ([6], [7]).

In order to avoid the inconsistency of performance metrics, this paper presents an optimal control based approach to address the path planning and trajectory planning problems simultaneously. The presented approach is able to generate time optimal trajectories without predetermining the geometric path while satisfying constraints involving robot dynamics. Similar works can be found in [8], [9], [10]. However either robot dynamics are ignored or long computation time is required in these works. In this paper, an efficient numerical method for trajectory optimization is utilized to solve the optimal control problem for robot motion planning. It is shown by numerical results that the solution can be found with short computation time even when complicated robot dynamics are involved. Experimental results have shown the feasibility of the planned motion.

The rest part of this paper is organized as follows: section II presents optimal control formulation for robot motion planning problems, section III presents an efficient numerical method for trajectory optimization, section IV presents numerical and experimental results of the proposed approach, and section V concludes this paper.

Yu Zhao, Hsien-Chung Lin, and Masayoshi Tomizuka are with the Department of Mechanical Engineering, University of California at Berkeley, Berkely, CA 94720, USA {yzhao334,hclin}@berkeley.edu, tomizuka@me.berkeley.edu

II. PROBLEM FORMULATION

A general optimal control problem can be posed as follows: determine the state-control function pair, $t \mapsto (\boldsymbol{x}, \boldsymbol{u})$, terminal time t_f, that minimize the performance metric or *cost function*, while satisfying *dynamic constraints*, *path constraints*, and *boundary conditions* ([11]). The robot motion planning problem can be formulated as an optimal control problem by defining the cost function, dynamic constraints, path constraints, and boundary conditions.

The state and control in motion planning involving robot dynamics can be defined as:

$$\boldsymbol{x}(t) = \begin{bmatrix} \boldsymbol{q}(t) \\ \dot{\boldsymbol{q}}(t) \end{bmatrix}, \quad \boldsymbol{u}(t) = \boldsymbol{\tau}(t) \tag{1}$$

where $t \in [0, t_f]$, $\boldsymbol{q}(t) = [q_1(t), \cdots, q_n(t)]^T$ is the vector for joint positions, and $\boldsymbol{\tau}(t) = [\tau_1(t), \cdots, \tau_n(t)]^T$ is the vector for joint torques, n is the number of robot joints.

A. Cost Function

The quality of the planned motion strongly depends on the formulation of cost function. In this paper, the cost function is formulated as a summation of motion time t_f and a regularization term for smoothness and naturalness of the generated motion:

$$J = t_f + \mu \int_0^{t_f} \dddot{\boldsymbol{q}}(t)^T \boldsymbol{Q} \dddot{\boldsymbol{q}}(t) \mathrm{d}t \tag{2}$$

where $\dddot{\boldsymbol{q}}(t)$ is the jerk of joint motion. The regularization term is designed based on the minimum-jerk model of human motion ([12]) and thus corresponds to the importance of the naturalness of the generated motion. $\mu \geq 0$ is a weighting coefficient for the regulation term, and $\boldsymbol{Q}$ is a weight matrix designed to penalize the motion of joints with higher gear ratios. The weight matrix $\boldsymbol{Q}$ is defined as

$$\boldsymbol{Q}(i,j) = \begin{cases} 0, & j \neq i \\ 1 \,/ \boldsymbol{R}(i,i)^2, & j = i \end{cases} \tag{3}$$

The case $\mu = 0$ corresponds to the time optimal motion planning problem. Larger μ slows down the generated motion, but increases the naturalness and smoothness.

B. Dynamic Constraints

The dynamic constraints is the robot dynamics. The equations of motion can be derived using Lagrangian's equations or Newton-Euler approach:

$$\boldsymbol{M}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \boldsymbol{G}(\boldsymbol{q}) + \boldsymbol{f}^f = \boldsymbol{\tau} \tag{4}$$

where $\boldsymbol{M}(\boldsymbol{q}) \in \mathbb{R}^{n\times n}$ is the inertia matrix, $\boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} \in \mathbb{R}^{n\times 1}$ is the Coriolis and centrifugal force term, $\boldsymbol{G}(\boldsymbol{q}) \in \mathbb{R}^{n\times 1}$ is the gravity term, and $\boldsymbol{f}^f \in \mathbb{R}^{n\times 1}$ is the friction term. For multi-jointed robot arms, all of these terms are inherently nonlinear.

Letting $\boldsymbol{x}_1 = \boldsymbol{q}$, $\boldsymbol{x}_2 = \dot{\boldsymbol{q}}$, the dynamic constraints can be formulated as state space model with state $\boldsymbol{x} = [\boldsymbol{x}_1^T, \boldsymbol{x}_2^T]^T$ and control $\boldsymbol{u} = \boldsymbol{\tau}$:

$$\begin{cases} \dot{\boldsymbol{x}}_1 = \boldsymbol{x}_2 & \text{(5a)} \\ \dot{\boldsymbol{x}}_2 = \boldsymbol{M}(\boldsymbol{x}_1)^{-1}\left[\boldsymbol{u} - \boldsymbol{C}(\boldsymbol{x}_1, \boldsymbol{x}_2)\boldsymbol{x}_2 - \boldsymbol{G}(\boldsymbol{x}_1) - \boldsymbol{f}^f\right] & \text{(5b)} \end{cases}$$

The state space model can be rewritten as:

$$\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{x} = \boldsymbol{F}(\boldsymbol{x}, \boldsymbol{u}) \tag{6}$$

C. Path Constraints

A set of path constraints can be formulated to accommodate various physical limitations of the robot actuators, as well as collision free conditions. The path constraints for robot motion planning include:

$$\begin{aligned} &\text{Position bounds:} && \boldsymbol{q}_{min} \le \boldsymbol{q}(t) \le \boldsymbol{q}_{max} && \text{(7a)} \\ &\text{Velocity bounds:} && \dot{\boldsymbol{q}}_{min} \le \dot{\boldsymbol{q}}(t) \le \dot{\boldsymbol{q}}_{max} && \text{(7b)} \\ &\text{Torque bounds:} && \boldsymbol{\tau}_{\min} \le \boldsymbol{\tau}(t) \le \boldsymbol{\tau}_{\max} && \text{(7c)} \\ &\text{Torque rate bounds:} && \dot{\boldsymbol{\tau}}_{\min} \le \dot{\boldsymbol{\tau}}(t) \le \dot{\boldsymbol{\tau}}_{\max} && \text{(7d)} \end{aligned}$$

Collision free conditions are also included in the path constraints. Dues to the complicated geometric mapping between robot workspace and configuration space, it is difficult to represent collision free conditions analytically. To simplify the formulation, the robot links and obstacles can be approximated by a set of spheres for differentiable collision detection, as illustrated in Fig. 1. The approximation can be performed either manually or automatically using sphere-tree construction algorithms [13]. Suppose M spheres are used to approximate robot links, and S spheres are used to approximate obstacles. Let the center and radius of each sphere that representing robot links be $\boldsymbol{c}_j^{rob}(\boldsymbol{q}(t)), r_j^{rob}, j = 1, \cdots, M$, the center and radius of each sphere that representing obstacles be $\boldsymbol{c}_k^{obs}, r_k^{obs}, k = 1, \cdots, S$. Robot forward kinematics problem can be solved to determine the functional relationship between joint positions $\boldsymbol{q}(t)$ and the location of sphere centers $\boldsymbol{c}_j^{rob}(\boldsymbol{q}(t))$. The collision free constraints can then be formulated as

$$\begin{aligned} &\text{Self:} && \|\boldsymbol{c}_j^{rob} - \boldsymbol{c}_k^{rob}\|_2 \ge r_j^{rob} + r_k^{rob},\ [j,k] \in I && \text{(8a)} \\ &\text{Obstacle:} && \|\boldsymbol{c}_j^{rob} - \boldsymbol{c}_l^{obs}\|_2 \ge r_j + r_l^{obs},\ \forall j, l && \text{(8b)} \end{aligned}$$

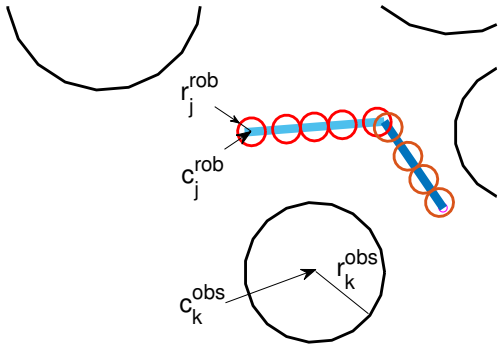


Fig. 1: Sphere approximation of robot and obstacles

where I is a set of indices indicating possible collision between two balls that approximate robot links.

When workspace boundaries are presented, additional path constraints are necessary. Let $[x_{\min}^{bnd}, x_{\max}^{bnd}]$, $[y_{\min}^{bnd}, y_{\max}^{bnd}]$, and $[z_{\min}^{bnd}, z_{\max}^{bnd}]$ be the workspace limits for X, Y, and Z directions respectively. Let $\boldsymbol{c}_{\min} = [x_{\min}^{bnd}, y_{\min}^{bnd}, z_{\min}^{bnd}]^T$, $\boldsymbol{c}_{\max} = [x_{\max}^{bnd}, y_{\max}^{bnd}, z_{\max}^{bnd}]^T$. The path constraints for workspace boundary can be formulated as:

$$\begin{aligned} &\text{Workspace lower bound:} && \boldsymbol{c}_{\min} \le \boldsymbol{c}_j^{rob} - r_j^{rob},\ \forall j && \text{(9a)} \\ &\text{Workspace upper bound:} && \boldsymbol{c}_{\max} \ge \boldsymbol{c}_j^{rob} + r_j^{rob},\ \forall j && \text{(9b)} \end{aligned}$$

D. Boundary Conditions

The boundary conditions for robot motion planning problem include:

$$\begin{aligned} &\text{Initial \& final position} && \boldsymbol{q}(0) = \boldsymbol{q}^0, \boldsymbol{q}(t_f) = \boldsymbol{q}^f && \text{(10a)} \\ &\text{Initial \& final velocity} && \dot{\boldsymbol{q}}(0) = 0, \dot{\boldsymbol{q}}(t_f) = 0 && \text{(10b)} \\ &\text{Initial \& final acceleration} && \ddot{\boldsymbol{q}}(0) = 0, \ddot{\boldsymbol{q}}(t_f) = 0 && \text{(10c)} \\ &\text{Terminal time bounds :} && t_f^{\min} \le t_f \le t_f^{\max} && \text{(10d)} \end{aligned}$$

where $\boldsymbol{q}^0$ and $\boldsymbol{q}^f$ are the initial and target joint positions, $t_f^{\min}$ and $t_f^{\max}$ are the minimum and maximum allowed terminal time.

III. Efficient Numerical Method for Trajectory Optimization

Trajectory optimization is a technique for computing an open-loop solution to an optimal control problem. Since no universal analytical solution can be found for nonlinear optimal control problems, a variety of numerical approaches have been developed for trajectory optimization in [14], [15]. In most numerical approaches, the continuous time optimal control problem is firstly converted into discretized optimization problem in a procedure called transcription ([16]). The optimization problem is then solved by general purpose optimization solver. Polynomial interpolation is finally utilized to return an approximate solution to the continuous time optimal control problem.

In numerical methods for trajectory optimization, two parts are playing the key role: discretization and optimization. An efficient implementation can be designed by choosing

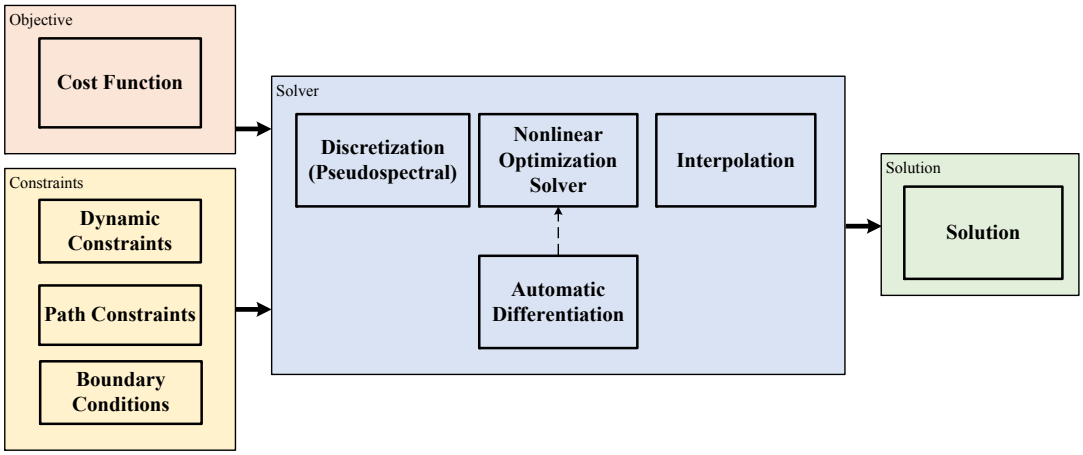


Fig. 2: Efficient numerical method for trajectory optimization

these two components intelligently. In this paper, the pseudospectral method is chosen to transcribe the continuous time optimal control problem, and the interior point method with the support of automatic differentiation is chosen to solve the discretized optimization problem, as illustrated in Fig. 2.

A. Pseudospectral Method

The decision variables in the continuous time optimal control problem for robot motion planning include t_f, $\boldsymbol{x}(t)$, and $\boldsymbol{u}(t)$. In the transcription procedure, $\boldsymbol{x}(t)$ and $\boldsymbol{u}(t)$ are discretized by their values at certain time as $\{\boldsymbol{x}(T_i), i = 0, \cdots N\}$ and $\{\boldsymbol{u}(T_i), i = 0, \cdots N\}$, where $\{T_i, i = 0, \cdots N\}$ are called knots, and N is the number of knots. It is reported in previous research [14], [17], [18] that the solution accuracy increases exponentially fast with the increase of interpolation knots for pseudospectral methods. Thus high computational efficiency can be achieved using pseudospectral method since less discretization knots can be chosen under the same solution accuracy requirement. In addition, the approximate solution is guaranteed to be smooth since high order global polynomial interpolation is utilized in pseudospectral methods.

In this paper, the Chebyshev-Lobatto points (or Chebyshev points) are chosen to be the knots. Such choice can avoid the oscillation phenomenon in high order global polynomial interpolation. For $t \in [0, t_f]$, the knots are:

$$T_i = \frac{t_f}{2}\left[\cos\left(\frac{i\pi}{N}\right) + 1\right], i = 0, \cdots, N \tag{11}$$

Pseudospectral methods have provided a set of tools for polynomial interpolation, approximating integration terms using quadrature, and approximating derivatives using differential matrix.

1) **Interpolation** The polynomial interpolation in pseudospectral methods can be performed by *barycentric interpolation*, which can be formulated as a linear combination of Lagrangian polynomials. For state trajectory and control trajectory, the form is

$$\begin{aligned} \boldsymbol{x}(t) &\approx \sum_{j=0}^{N} \boldsymbol{x}(T_j)\ell_j(t) \\ \boldsymbol{u}(t) &\approx \sum_{j=0}^{N} \boldsymbol{u}(T_j)\ell_j(t) \end{aligned} \tag{12}$$

where $\ell_j(t)$ is the jth Lagrange polynomial. In barycentric interpolation, a special form of Lagrange polynomial is implemented to efficiently perform interpolation ([18], [19]).

2) **Quadrature** Quadrature is the standard term for the numerical calculation of integrals. The integration of function $\mathcal{L}[\boldsymbol{x}]$ can be approximately evaluated by quadrature rules in pseudospectral methods as:

$$\begin{aligned} \int_0^{t_f} \mathcal{L}[\boldsymbol{x}]\mathrm{d}t &\approx \int_0^{t_f} \left[\sum_{j=0}^{N} \ell_j(t)\mathcal{L}[\boldsymbol{x}(T_j)]\right] \mathrm{d}t \\ &= \sum_{j=0}^{N} w_j \mathcal{L}[\boldsymbol{x}(T_j)] \end{aligned} \tag{13}$$

where $\{w_j, j = 0, \cdots, N\}$ is a set of quadrature weights. The quadrature weights can be explicitly defined to be

$$w_j = \int_0^{t_f} \ell_j(t)\mathrm{d}t, j = 0, \cdots, N \tag{14}$$

When Chebyshev points are chosen, the corresponding quadrature rule (Clenshaw–Curtis quadrature) can be found in [20]:

3) **Differentiation matrix** Let the stacked state and robot dynamics at knots be

$$\boldsymbol{X} = \begin{bmatrix} \boldsymbol{x}(T_0)^T \\ \vdots \\ \boldsymbol{x}(T_N)^T \end{bmatrix}, \boldsymbol{\mathcal{F}} = \begin{bmatrix} \boldsymbol{F}(\boldsymbol{x}(T_0), \boldsymbol{u}(T_0))^T \\ \vdots \\ \boldsymbol{F}(\boldsymbol{x}(T_N), \boldsymbol{u}(T_N))^T \end{bmatrix} \tag{15}$$

The dynamic constraints can be posed as [14]:

$$\boldsymbol{D}\boldsymbol{X} = \frac{t_f}{2}\boldsymbol{\mathcal{F}} \tag{16}$$

where $\boldsymbol{D}$ is the differential matrix that is used to compute the scaled time derivative of the polynomial approximation of $\boldsymbol{x}$ ([19]). Let the stacked joint torques be $\boldsymbol{U} = [\boldsymbol{u}(T_0), \cdots, \boldsymbol{u}(T_N)]^T$, the stacked torque rates $\boldsymbol{U}_d = [\dot{\boldsymbol{u}}(T_0), \cdots, \dot{\boldsymbol{u}}(T_N)]^T$ can be approximately calculated as:

$$\boldsymbol{U}_d \approx \frac{2}{t_f}\boldsymbol{D}\boldsymbol{U} \tag{17}$$

B. Automatic Differentiation

Lots of optimization solvers are based on gradient descent algorithm. Derivative of objective function and constraints are frequently evaluated by numerical differentiation approaches, which perturbs input to the function in each dimension to obtain an approximation of the derivative using finite differences. However, numerical differentiation approaches are computationally expensive for functions with high dimensional input, and inevitably introduces round-off errors. Symbolic differentiation is one way to avoid round-off errors, however it frequently leads to inefficient code. Both numerical differentiation and symbolic differentiation are problematic in the calculation of higher order derivatives like Hessian.

To address the problems in numerical differentiation and symbolic differentiation, automatic differentiation is introduced. Automatic differentiation is a set of techniques to evaluate derivative of a function ([21]). The computational cost of automatic differentiation is lower than numerical differentiation or symbolic differentiation. A rich collection of automatic differentiation implementations can be found in [22], [23], [24]. In this paper, CasADi [25] is chosen for its good usability in MATLAB environment. Since computational cost of automatic differentiation is proportional to that for function evaluation, articulated body algorithm [26] is utilized in this work for efficient evaluation of robot dynamics.

IV. Numerical and Experimental Results

Motion planning of a 6-axis industrial robot with dynamic constraints is considered as an example. The industrial robot is supposed to work in a constrained workspace for pick-and-place tasks. The geometric model of the industrial robot is shown in Fig. 3. The path constraints include joint position, velocity, torque, torque rate bounds, and collision free conditions. The sphere approximation of robot links, obstacles, and workspace boundary is shown in Fig. 4. The actuator limits are listed in Table I. The actuator limitations are designed to be conservative for safety.

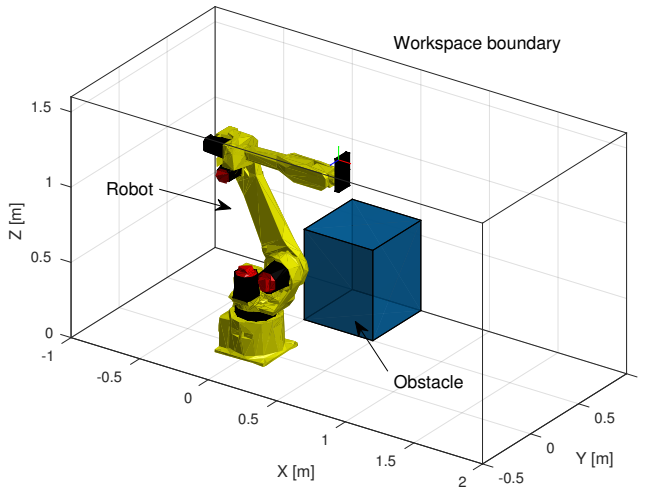


Fig. 3: Geometric model of a 6-axis industrial robot

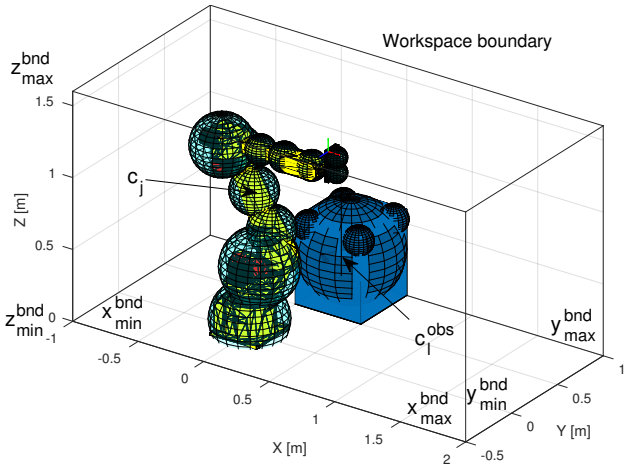


Fig. 4: Sphere approximation of 6-axis robot and obstacle

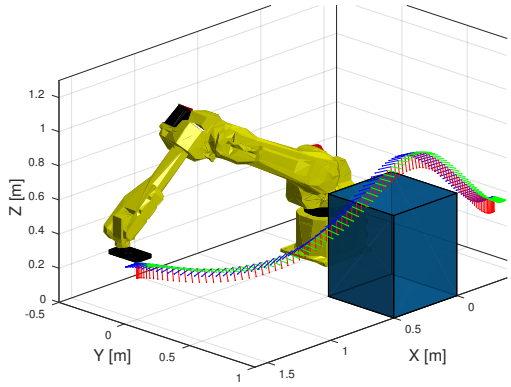
TABLE I: Actuator limits of 6-axis industrial robot

Limits	J1	J2	J3	J4	J5	J6
$\boldsymbol{q}_{\min}[^\circ]$	-160	-90	-120	-180	-120	-180
$\boldsymbol{q}_{\max}[^\circ]$	170	90	230	180	100	180
$\dot{\boldsymbol{q}}[^\circ/s]$	±165	±165	±175	±350	±340	±520
$\boldsymbol{\tau}$[Nm]	±1397	±1402	±383	±45.2	±44.6	±32.5
$\dot{\boldsymbol{\tau}}$[Nm/s]	±20948	±21035	±5741	±678	±669	±488

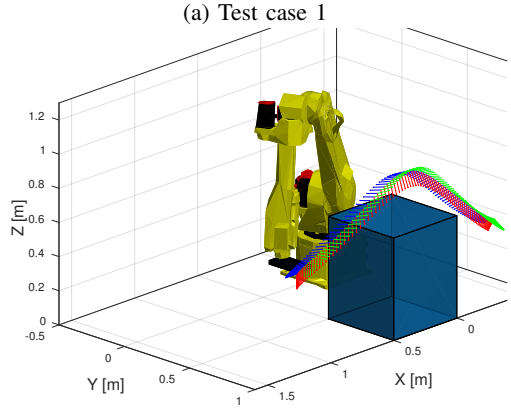
A. Numerical Results

Several tests have been performed to evaluate the effect of different regularization weights and number of knots. It is observed that the geometric path can be adjusted automatically for collision avoidance, even if infeasible initialization is provided. Increasing μ is helpful for shortening the computation time, but results in slower motion. Shorter motion time t_f can be obtained by using more knots, but the computation takes longer time. When 12 knots are chosen, the planned motion is reasonably close to the possible optimal solution with an acceptable computation time. Less knots can be used if shorter computation time is required. $\mu = 0.3$ can be chosen to balance the time-optimality requirement and the naturalness of the generated motion.

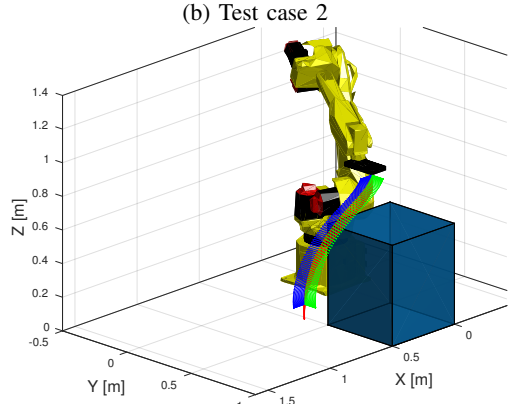
The proposed planning algorithm has also been tested using a group of different initial and target positions, as shown in Fig. 5. All the computation is performed on a laptop with a 2.1 GHz Intel® Core™ processor. The solver takes around 5.9 s for a one-time initialization for automatic differentiation by CasADi. The computation times and optimized motion times for the test cases are summarized



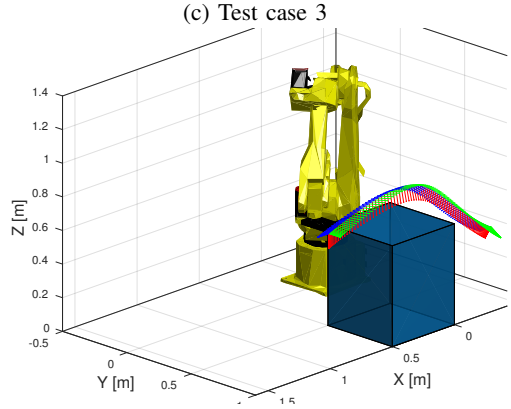
(a) Test case 1



(b) Test case 2



(c) Test case 3



(d) Test case 4

Fig. 5: Optimal multiple joint robot trajectory with different initial and target positions, 12 knots, $\mu = 0.3$

TABLE II: Motion time and computation time for test cases

time [s]	knots	case 1	case 2	case 3	case 4
motion time t_f	12	1.65	1.23	1.05	1.08
	8	1.98	1.23	1.09	1.11
computation time	12	2.55	2.37	2.39	2.57
	8	1.28	1.75	0.84	1.32

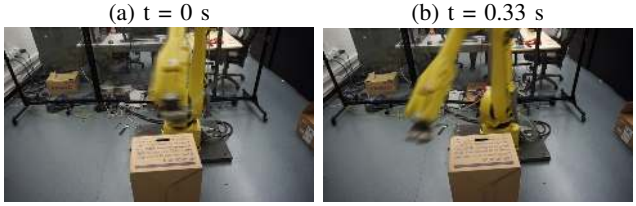
in Table II. In all the test cases, good approximations of the time optimal trajectories are returned in about 2-3 seconds using 12 knots, and about 1-2 seconds using 8 knots. Existing works [9], [10] require from 20 seconds to several minutes for computation, in which only robot dynamics are involved but not collision avoidance. The proposed approach is highly efficient comparing to these results.

B. Experimental Results

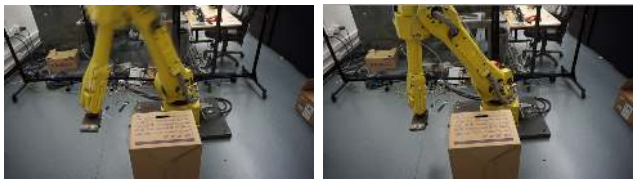
The planned optimal trajectory of test case 2 in Fig. 5 has been used as motion reference in experiment at the Mechanical Systems Control laboratory at the University of California, Berkeley. The actual robot motions are captured from video record as shown in Fig. 6. As shown in the figure, the planned motion is collision free.

The scaled joint velocities and joint torques are shown in Fig. 7. As shown in the figure, the planned motion is feasible under conservative actuator limitations. The motion time t_f is adjusted automatically from an initial value 10 s to about 1.2s. It is also observed that the planned motion is close to time-optimal with at least one of the constraints is active.

(a) t = 0 s (b) t = 0.33 s



(c) t = 0.56 s (d) t = 0.8 s

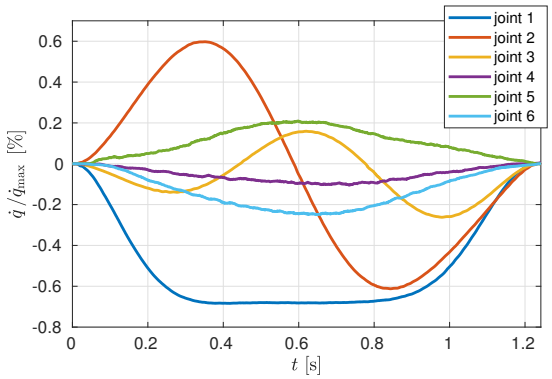


(e) t = 0.96 s (f) t = 1.2 s

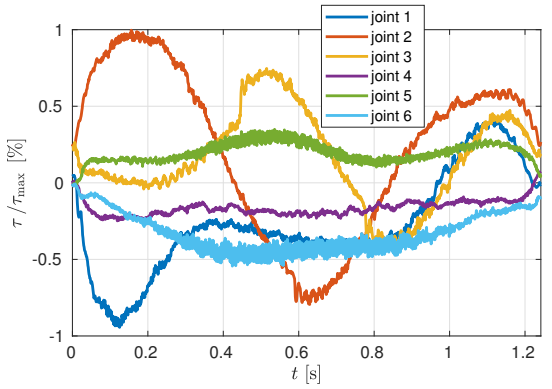
Fig. 6: Actual robot motion in experiment

V. Conclusion

Motion planning involving complicated robot dynamics and geometric constraints is challenging. Since most approaches decompose motion planning to two subtopics and



(a) Joint velocity



(b) Joint torque

Fig. 7: Measured joint velocity and torque of optimal robot trajectory in experiment

deal with them separately, only suboptimal solution can be found. This paper presents an optimal control based approach to address the path planning and trajectory planning problems simultaneously. An efficient numerical method for trajectory optimization is proposed as one practical solution for the nonlinear optimal control problem. Numerical results have shown that the motion planning problem can be solved with a short computation time and reasonable accuracy. Experimental results have verified the effectiveness and feasibility of the planning algorithm. It is worth investigating improvements to this approach and exploring possibilities to implement it in different robotic applications.

REFERENCES

[1] Q.-C. Pham, S. Caron, P. Lertkultanon, and Y. Nakamura, "Admissible velocity propagation: Beyond quasi-static path planning for high-dimensional robots," *The International Journal of Robotics Research*, vol. 36, no. 1, pp. 44–67, 2017.

[2] S. M. LaValle, *Planning algorithms*. Cambridge university press, 2006.

[3] I. A. Sucan, M. Moll, and L. E. Kavraki, "The open motion planning library," *IEEE Robotics & Automation Magazine*, vol. 19, no. 4, pp. 72–82, 2012.

[4] D. Verscheure, B. Demeulenaere, J. Swevers, J. De Schutter, and M. Diehl, "Time-optimal path tracking for robots: A convex optimization approach," *IEEE Transactions on Automatic Control*, vol. 54, no. 10, pp. 2318–2327, 2009.

[5] P. Reynoso-Mora, W. Chen, and M. Tomizuka, "A convex relaxation for the time-optimal trajectory planning of robotic manipulators along predetermined geometric paths," *Optimal Control Applications and Methods*, vol. 37, no. 6, pp. 1263–1281, 2016.

[6] S. M. La Valle, "Motion planning," *IEEE Robotics & Automation Magazine*, vol. 18, no. 2, pp. 108–118, 2011.

[7] Q.-C. Pham, "A general, fast, and robust implementation of the time-optimal path parameterization algorithm," *IEEE Transactions on Robotics*, vol. 30, no. 6, pp. 1533–1540, 2014.

[8] J. Schulman, J. Ho, A. X. Lee, I. Awwal, H. Bradlow, and P. Abbeel, "Finding locally optimal, collision-free trajectories with sequential convex optimization." in *Robotics: science and systems*, vol. 9, no. 1. Citeseer, 2013, pp. 1–10.

[9] T. Chettibi, H. Lehtihet, M. Haddad, and S. Hanchi, "Minimum cost trajectory planning for industrial robots," *European Journal of Mechanics-A/Solids*, vol. 23, no. 4, pp. 703–715, 2004.

[10] M. Diehl, H. G. Bock, H. Diedam, and P.-B. Wieber, "Fast direct multiple shooting algorithms for optimal robot control," in *Fast motions in biomechanics and robotics*. Springer, 2006, pp. 65–93.

[11] I. M. Ross and M. Karpenko, "A review of pseudospectral optimal control: From theory to flight," *Annual Reviews in Control*, vol. 36, no. 2, pp. 182–197, 2012.

[12] T. Flash and N. Hogan, "The coordination of arm movements: an experimentally confirmed mathematical model," *Journal of neuroscience*, vol. 5, no. 7, pp. 1688–1703, 1985.

[13] G. Bradshaw and C. O'Sullivan, "Adaptive medial-axis approximation for sphere-tree construction," *ACM Transactions on Graphics (TOG)*, vol. 23, no. 1, pp. 1–26, 2004.

[14] A. V. Rao, "A survey of numerical methods for optimal control," *Advances in the Astronautical Sciences*, vol. 135, no. 1, pp. 497–528, 2009.

[15] J. T. Betts, *Practical methods for optimal control and estimation using nonlinear programming*. Siam, 2010, vol. 19.

[16] M. P. Kelly, "Transcription methods for trajectory optimization," *Tutorial, Cornell University, Feb*, 2015.

[17] M. Kelly, "An introduction to trajectory optimization: How to do your own direct collocation," *SIAM Review*, vol. 59, no. 4, pp. 849–904, 2017.

[18] L. N. Trefethen, *Approximation theory and approximation practice*. Siam, 2013, vol. 128.

[19] J.-P. Berrut and L. N. Trefethen, "Barycentric lagrange interpolation," *SIAM review*, vol. 46, no. 3, pp. 501–517, 2004.

[20] J. Waldvogel, "Fast construction of the fejér and clenshaw–curtis quadrature rules," *BIT Numerical Mathematics*, vol. 46, no. 1, pp. 195–202, 2006.

[21] A. Griewank and A. Walther, *Evaluating derivatives: principles and techniques of algorithmic differentiation*. Siam, 2008, vol. 105.

[22] A. Walther and A. Griewank, "Getting started with adol-c." in *Combinatorial scientific computing*, 2009, pp. 181–202.

[23] B. M. Bell, "Cppad: a package for c++ algorithmic differentiation," *Computational Infrastructure for Operations Research*, vol. 57, 2012.

[24] C. Bendtsen and O. Stauning, "Fadbad, a flexible c++ package for automatic differentiation," Technical Report IMM–REP–1996–17, Department of Mathematical Modelling, Technical University of Denmark, Lyngby, Denmark, Tech. Rep., 1996.

[25] J. A. E. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl, "CasADi – A software framework for nonlinear optimization and optimal control," *Mathematical Programming Computation*, In Press, 2018.

[26] R. Featherstone, *Rigid body dynamics algorithms*. Springer, 2014.