An Optimized Path Planning of Manipulator with Spline Curves Using Real Quantifier Elimination Based on Comprehensive Gröbner Systems*

Yusuke Shirato[1], Natsumi Oka[1], Akira Terui[2,*] and Masahiko Mikawa[3]

[1]Master's Program in Mathematics, Graduate School of Science and Technology, University of Tsukuba, Tsukuba 305-8571, Japan
[2]Institute of Pure and Applied Sciences, University of Tsukuba, Tsukuba 305-8571, Japan
[3]Institute of Library, Information and Media Science, University of Tsukuba, Tsukuba 305-8550, Japan

Abstract

This paper presents an advanced method for addressing the inverse kinematics and optimal path planning challenges in robot manipulators. The inverse kinematics problem involves determining the joint angles for a given position and orientation of the end-effector. Furthermore, the path planning problem seeks a trajectory between two points. Traditional approaches in computer algebra have utilized Gröbner basis computations to solve these problems, offering a global solution but at a high computational cost. To overcome the issue, the present authors have proposed a novel approach that employs the Comprehensive Gröbner System (CGS) and CGS-based quantifier elimination (CGS-QE) methods to efficiently solve the inverse kinematics problem and certify the existence of solutions for trajectory planning. This paper extends these methods by incorporating smooth curves via cubic spline interpolation for path planning and optimizing joint configurations using shortest path algorithms to minimize the sum of joint configurations along a trajectory. This approach significantly enhances the manipulator's ability to navigate complex paths and optimize movement sequences.

Keywords

Gröbner basis, Comprehensive Gröbner Systems, Quantifier Elimination, Robotics, Inverse kinematics problem, Path planning, Trajectory planning

1. Introduction

This paper discusses a method for solving the inverse kinematics problem and the optimal path planning problem for a robot manipulator. A manipulator is a robot with links corresponding to human arms and joints corresponding to human joints, and the tip is called the end-effector. The inverse kinematics problem for manipulators is to find the angle of each joint, given the position and orientation of the end-effector. The path planning problem is to find a path to move the end-effector between two specified positions [1].

When operating the manipulator, one needs to solve the inverse kinematics problem (or the path planning problem, respectively) for the desired end-effector position (or the series of positions, respectively).

Methods of solving inverse kinematics problems for manipulators by reducing the inverse kinematics problem to a system of polynomial equations and using the Gröbner basis has been proposed [2, 3, 4, 5, 6]. Solving the inverse kinematics problem using the Gröbner basis computation has an advantage that the global solution of the inverse kinematics problem can be obtained before the end-effector will actually be "moved" by simulation or other means. On the other hand, the Gröbner basis computation has the disadvantage of relatively high computational cost compared to local solution methods such as the Newton method. Furthermore, when solving a path planning problem using the Gröbner basis

SCSS 2024: 10th International Symposium on Symbolic Computation in Software Science, August 28–30, 2024, Tokyo, Japan
*This work was partially supported by JKA and its promotion funds from KEIRIN RACE.
*Corresponding author.
terui@math.tsukuba.ac.jp (A. Terui); mikawa@slis.tsukuba.ac.jp (M. Mikawa)
https://researchmap.jp/aterui (A. Terui); https://mikawalab.org/ (M. Mikawa)
0000-0003-0846-3643 (A. Terui); 0000-0002-2193-3198 (M. Mikawa)

computation, it is necessary to solve the inverse kinematics problem for each point on the path, which is even more computationally expensive.

The third and fourth authors have also previously proposed methods for solving inverse kinematics problems using Gröbner basis computations [7, 8, 9]. The authors' contributions in their previous work [9] are as follows. We have proposed a method for solving the inverse kinematics problem and the trajectory planning problem of a 3 Degree-Of-Freedom (DOF) manipulator using the Comprehensive Gröbner System (CGS) [10]. In the proposed method, the inverse kinematic problem has been expressed as a system of polynomial equations with the coordinates of the end-effector as parameters and the CGS is calculated in advance to reduce the cost of Gröbner basis computation for each point on the path. In addition, we have proposed a method for solving the inverse kinematics problems using the CGS-QE method [11], which is a quantifier elimination (QE) method based on CGS computations, to certify the existence of a (real) solution. Furthermore, we have proposed a method to certify the existence of a solution to the whole trajectory planning problem using the CGS-QE method, where the points on the trajectory are represented by parameters.

This paper proposes the following method as an extension of the previous work [9].

1. Extension of paths used in path planning problems: while the previous work has used straight lines, this paper uses smooth curves generated by the cubic spline interpolation passing through given points. Using a curved path allows us to plan paths that avoid obstacles, for example.
2. Optimization of the joint configuration obtained as the solution to the path planning problem: when solving the path planning problem, there can be multiple solutions to the inverse kinematics problem at each point on the path. In this case, the question is which of the solutions for adjacent points can be connected to minimize the sum of the configurations of the entire sequence of the joints. In this paper, we reduce this problem to the shortest path problem of a weighted graph and propose a method to compute the optimal sequence of joint configurations using shortest path algorithms.

2. Solving path planning problems in 3-DOF manipulators

The manipulator used in this paper is myCobot 280 [12] from Elephant Robotics, Inc. (hereafter referred to simply as "myCobot"). Although myCobot is a 6-DOF manipulator, we treat it as a 3-DOF manipulator by operating only the three main joints used to move the end-effector while the remaining joints are fixed, due to the computational costs for solving the inverse kinematic problem by using the CGS and the CGS-QE method.

2.1. Formulation of the forward and the inverse kinematics problems

Figure 1 shows the components and the coordinate systems of myCobot. The forward kinematics problem for myCobot is derived using the modified Denavit-Hatenberg convention (hereafter abbreviated as "D-H convention"), which is standard in robotics [1]. Let be the Cartesian coordinate system with the origin at the manipulator's base, and let (x, y, z) be the coordinates of the end-effector in . Let $_i$ be the coordinate system with the origin at the joint i. Note that Joint 0 represents the base and Joint 8 represents the end-effector.

Let $\theta_1, \theta_3, \theta_4$ be the configuration of the joints 1, 3, 4, respectively, to be operated in this paper, and let $s_i = \sin\theta_i$, $c_i = \cos\theta_i$ for $i = 1, 3, 4$. In the following, for a set of polynomials $F = \{f_1, \dots, f_m\} \subset \mathbb{R}[c_1, s_1, c_3, s_3, c_4, s_4]$, The system of polynomial equations $f_1 = \cdots = f_m = 0$ is denoted by $F = 0$. In this case, the inverse kinematics problem is to find the solution $c_1, s_1, c_3, s_3, c_4, s_4$ to the system of polynomial

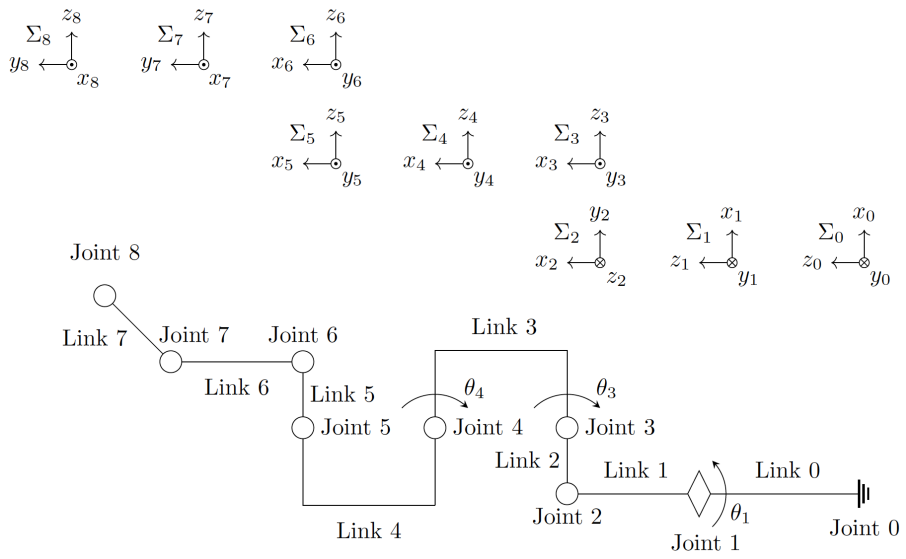

Figure 1: Components and the coordinate systems of myCobot.

equations $F = 0$ with $F = \{f_1, \dots, f_6\}$, where

$$\begin{aligned} f_1 &= -16918 s_1 s_3 c_4 - 16918 s_1 c_3 s_4 - 4360 s_1 s_3 s_4 + 4360 s_1 c_3 c_4 - 11040 s_1 s_3 \\ &\quad - 6639 c_1 + 100 x, \\ f_2 &= 16918 c_1 s_3 c_4 + 16918 c_1 c_3 s_4 + 4360 c_1 s_3 s_4 - 4360 c_1 c_3 c_4 + 11040 c_1 s_3 \\ &\quad - 6639 s_1 + 100 y, \\ f_3 &= -16918 c_3 c_4 + 16918 s_3 s_4 - 4360 c_3 s_4 - 4360 s_3 c_4 - 11040 c_3 \\ &\quad - 13156 + 100 z, \\ f_4 &= s_1^2 + c_1^2 - 1, \quad f_5 = s_3^2 + c_3^2 - 1, \quad f_6 = s_4^2 + c_4^2 - 1. \end{aligned} \tag{1}$$

Note that the system of equations $F = 0$ has parameters x, y, z in the coefficients.

With our previously proposed method [9], before solving $F = 0$, we calculate the CGS of $\langle f_1, \dots, f_6 \rangle$. Then, for a given end-effector coordinate $(x, y, z) = (\alpha, \beta, \gamma) \in \mathbb{R}^3$, determine the feasibility of such a configuration using the CGS-QE method. If the configuration is feasible, we solve $F = 0$ numerically after substituting parameters x, y, z with α, β, γ, respectively, to obtain the inverse kinematic solution.

2.2. Generating trajectories using cubic spline interpolation

In the path planning problem, a finite number of points through which the path passes are given in advance, and a trajectory is generated on the smooth path passing through these points. In our previous study, a straight line was used as the path, but in this paper, a path is generated using cubic spline interpolation [13].

Let (x_0, y_0, z_0), (x_1, y_1, z_1), (x_2, y_2, z_2), (x_3, y_3, z_3) be four different points given in $\mathbb{R}^3$. For $j = 0, 1, 2$, calculate a curve C_j passing through (x_j, y_j, z_j) and $(x_{j+1}, y_{j+1}, z_{j+1})$ in the form of a function $(X_j(s), Y_j(s), Z_j(s))$ with $s \in [j, j+1]$ as a parameter. In cubic spline interpolation, $X_j(s), Y_j(s), Z_j(s)$ are cubic polynomials, respectively, satisfying the following conditions.

$$\begin{aligned} (X_j(j), Y_j(j), Z_j(j)) &= (x_j, y_j, z_j), \quad j = 0, 1, 2, \\ (X_j(j+1), Y_j(j+1), Z_j(j+1)) &= (x_{j+1}, y_{j+1}, z_{j+1}), \quad j = 0, 1, 2, \\ (X_j'(j+1), Y_j'(j+1), Z_j'(j+1)) &= (X_{j+1}'(j+1), Y_{j+1}'(j+1), Z_{j+1}'(j+1)), \quad j = 0, 1, \\ (X_j''(j+1), Y_j''(j+1), Z_j''(j+1)) &= (X_{j+1}''(j+1), Y_{j+1}''(j+1), Z_{j+1}''(j+1)), \quad j = 0, 1. \end{aligned} \tag{2}$$

Table 1
Test points for the inverse kinematics problem. The unit of the coordinates is [mm].

Test	(x_0, y_0, z_0)	(x_1, y_1, z_1)	(x_2, y_2, z_2)	(x_3, y_3, z_3)
1	$(-100, -100, 100)$	$(0, -150, 50)$	$(50, -100, 50)$	$(100, 0, 0)$
2	$(-150, -100, 150)$	$(0, -100, 100)$	$(100, -50, 50)$	$(100, 100, 0)$
3	$(-250, 0, 0)$	$(-150, 0, 50)$	$(-150, 100, 150)$	$(250, 100, 100)$
4	$(100, 50, 0)$	$(50, 100, 150)$	$(-150, 150, 100)$	$(-150, 50, 50)$
5	$(100, 100, -50)$	$(50, 100, 0)$	$(-100, 100, 50)$	$(-150, -50, 100)$
6	$(150, 150, 0)$	$(50, 50, 100)$	$(-50, -50, 100)$	$(-150, -150, 50)$

In this paper, in addition to the conditions in eq. (2), we find the *natural* cubic Spline interpolation that satisfies

$$X_0''(0) = X_3''(3) = Y_0''(0) = Y_3''(3) = Z_0''(0) = Z_3''(3) = 0. \tag{3}$$

2.3. Solving the inverse kinematics problem

For the trajectory C obtained in the previous section, we solve the inverse kinematics problem using the following procedure.

First, using the CGS-QE method, we determine the existence of solutions to the inverse kinematics problem for each curve C_j $(j = 0, 1, 2)$ that constitutes the path to move the end-effector. In eq. (1), replace x, y, z in each equation by $X_j(s), Y_j(s), Z_j(s)$, respectively, to obtain a system of polynomial equations with s as parameter. Let $F_j'(s)$ be the result.

Next, we apply the CGS-QE method to $F_j'(s)$ to determine whether the system of equations $F_j'(s) = 0$ has real solutions within the parameter $s \in [j, j+1]$. If $F_j'(s) = 0$ has real solutions within the parameter $s \in [j, j+1]$ (i.e., within $s \in [0, 3]$ throughout for s), we calculate the trajectory of the end-effector's position for time t. Let $T = 3u$ (where u is a positive integer) and, for time $t = 0, \ldots, T$, let $s = f(t)$ where f is a continuous function of $[0, T] \to [0, 3]$. To ensure that the end-effector has no velocity and acceleration at the beginning and end of the trajectory, we obtain f as a polynomial of the smallest degree possible to satisfy $f'(t) = f''(t) = 0$ at $t = 0$ and T. Then, we obtain $f(t) = 3\left(\frac{18t^5}{T^5} - \frac{45t^4}{T^4} + \frac{30t^3}{T^3}\right)$ [14].

Finally, we solve the inverse kinematics problem $t = 0, \ldots, T$. For $j = 0, 1, 2$ and $t = ju, \ldots, (j+1)u$, the configuration of the joints is obtained by solving the system of polynomial equations

$$F_j'(f(t)) = 0. \tag{4}$$

Assuming that the time required to solve the inverse kinematics problem (4) at each value of t is constant, then the computation time for trajectory planning is expected to be proportional to T.

2.3.1. Experimental results

We have implemented the above method and conducted experiments. The implementation of the inverse kinematics computation was performed as follows: the CGS computation was performed using an algorithm by Kapur et al. [15] with the implementation by Nabeshima [16] on the computer algebra system Risa/Asir [17]. The existence of the root of the inverse kinematics problem by the CGS-QE method was verified using Risa/Asir with accompanying Wolfram Mathematica 13.1 [18] for simplifying the expression.

We have given the set of test points as shown in Table 1, whose unit of the coordinates is [mm]. The experiments were conducted in the following environment: Intel Xeon Silver 4210 3.2 GHz, RAM 256 GB, Linux Kernel 5.4.0, Risa/Asir Version 20230315, Wolfram Mathematica 13.1.

Table 2 shows the results of the experiments for $T = 10$ and $T = 50$. The comprehensive Gröbner system for eq. (1) uses precomputed values. If any part of the generated spline for the given coordinates falls outside the feasible region of myCobot, an error is displayed. The average computation time does

Table 2
Results of path planning time in seconds.

Test	$T = 10$ [s]	$T = 50$ [s]
1	1.293	7.836
2	1.342	6.940
3	1.296	8.049
4	1.574	7.616
5	1.265	7.155
6	Fail	Fail
Average	1.354	7.429

not consider the computation time of tests that resulted in an error. The experimental results show that the computation time is considered to be approximately proportional to T. In Test 6, it appears that part of the path exceeds the feasible region, causing the computation to terminate with an error.

2.4. Solving optimal path planning problem

The system of equations (4) may have several different solutions at each time t. For time $t = 0, \dots, T$, suppose that there exist $k_{j,t}$ solutions to the configuration θ_j of joint j ($j = 1, 3, 4$) at t such that $\theta_{j,t}^{(1)}, \dots, \theta_{j,t}^{(k_{j,t})}$. The sum of the configuration changes of the joints from time $t = 1$ to T differs depending on the choice of the solution from $\theta_{j,t}^{(1)}, \dots, \theta_{j,t}^{(k_{j,t})}$ at each time t. In order to move each joint more smoothly, it is desirable to select a solution that minimizes the sum of the configuration changes of the joints on the path. For this purpose, for $j = 1, 3, 4$, we define a weighted graph $G_j = (V_j, E_j)$, where $V_j = \{\theta_{j,t}^{(k)} \mid t \in \{0, \dots, T\}, k \in \{1, \dots, k_{j,t}\}\}$ is the set of vertices corresponding to the joint configuration at each time t, and $E_j = \{(\theta_{j,t}^{(k_1)}, \theta_{j,t+1}^{(k_2)}) \mid 1 \leq k_1, k_2 \leq k_{j,t}, t \in \{0, \dots, T-1\}\}$ is the set of edges connecting the vertices at adjacent times. Then, in G_j, we reduce the problem of finding the optimal path to the one of finding the shortest path from $\theta_{j,0}^{(k)}$ to $\theta_{j,T}^{(k)}$. We have solved the shortest path problem using the following methods.

Method 1 For $t = 0, \dots, T-1$, find the sequence of joint configurations $\{\theta_{j,t}^{(k)} \mid t = 0, \dots, T\}$ satisfying the minimum distance between adjacent joint configurations such that $\min\{|\theta_{j,t}^{(k_1)} - \theta_{j,t+1}^{(k_2)}| \mid 1 \leq k_1 \leq k_{t,k_{j,t}}, 1 \leq k_2 \leq k_{t+1,k_{j,t+1}}\}$.

Method 2 In the graph G_j, find the shortest path starting at $\theta_{j,0}^{(k)}$ ($k = 1, \dots, k_{j,0}$) using the Dijkstra method [19], then find the shortest path with the minimum length among G_js.

The arithmetic complexity of each method above is estimated as follows. Let T be the number of points in the trajectory, and assume that the number of solutions to the inverse kinematics problem at each point is constant at d. The complexity of Method 1 is $O(Td)$. On the other hand, the complexity of Method 2, using a binary heap, is estimated to be $O(Td^2 \log(Td))$.

2.4.1. Experimental results

We have implemented the above method and conducted experiments. The implementation of each procedure was based on an implementation [20] in the Python programming language. The experiments were conducted in the following environment: A virtual machine with RAM 13.2 GB, Ubuntu 22.04.3 LTS, Python 3.10.2 on VMware Workstation 16 Player, on the host environment with Intel Core i7-1165G7, RAM 16 GB, Windows 11 Home.

The test points are composed of a total of $T = 15$ points, obtained by dividing each segment of the cubic spline curve passing through each point in Table 3 into 5 parts. The number of solutions to the inverse kinematics problem at each point in each test segment is $d = 4$.

Table 3
Test points for the optimal path planning problem. The unit of the coordinates is [mm].

Test	(x_0, y_0, z_0)	(x_1, y_1, z_1)	(x_2, y_2, z_2)	(x_3, y_3, z_3)
1	$(-100, -100, 100)$	$(0, -150, 50)$	$(50, -100, 50)$	$(100, 0, 0)$
2	$(-150, -100, 150)$	$(0, -100, 100)$	$(100, -50, 50)$	$(100, 100, 0)$
3	$(-250, 0, 0)$	$(-150, 0, 50)$	$(-150, 100, 150)$	$(250, 100, 100)$
4	$(100, 50, 0)$	$(50, 100, 150)$	$(-150, 150, 100)$	$(-150, 50, 50)$
5	$(100, 100, -50)$	$(50, 100, 0)$	$(-100, 100, 50)$	$(-150, -50, 100)$

Table 4
The sum of the joint variations [rad].

Test	Method 1 [rad]	Method 2 [rad]
1	8.3142	4.4013
2	11.5985	9.8986
3	15.0005	13.6731
4	8.5828	6.3277
5	7.1897	5.7108
Average	10.1371	8.0023

Table 5
Computing time of the optimal path computation [10^{-4}s].

Test	Method 1 [10^{-4}s]	Method 2 [10^{-4}s]
1	1.96	39.4
2	2.16	42.8
3	2.90	33.0
4	2.90	40.1
5	4.44	36.4
Average	2.87	31.3

Table 4 shows the sum of the joint variations [rad]. From the average values of each result, we see that Method 2 (with the Dijkstra method) reduces the total amount of joint rotation.

Table 5 shows the computation time for each method. From the average computation times, it can be seen that Method 1 is approximately 10 times faster than Method 2. Compared to the overall computation time for route planning shown in Table 2, the computation time for optimal route selection is considered sufficiently short, even for using Method 2 (Dijkstra method).

3. Concluding remarks

We have proposed a method for trajectory planning of robot manipulators using computer algebra, where the path is provided by cubic spline interpolation. Furthermore, we have proposed a method to optimize the joint configuration of the manipulator by solving the shortest path problem in a weighted graph. The experimental results have shown that the proposed methods can be used to plan the trajectory of the manipulator and optimize the joint configuration.

Future work includes the following.

1. Regarding path planning using cubic splines, it has been pointed out that some parts of the generated path may deviate from the feasible region. In the future, it is necessary to consider methods that ensure the generated trajectory stays within the feasible region while meeting given constraints, such as avoiding obstacles. In response to this, the authors are currently proposing a path-planning method that uses Bézier curves to generate trajectories that remain within the feasible region [21].

2. Instead of representing the trajectory on the path, it is desired to express any point on the path using parameters and ensure the solution to the inverse kinematics problem for that point. For linear paths, a method has already been proposed by the authors [9], and this will be extended to curves given by cubic splines.
3. Regarding optimal path planning using shortest path calculations on graphs, we have used an implementation of Dijkstra's algorithm using a binary heap in this paper. However, one method to further improve computational efficiency is to use Dijkstra's algorithm with a Fibonacci heap [22]. For example, using an implementation of Dijkstra's algorithm with a Fibonacci heap can be considered to improve computational efficiency.
4. To extend the proposed method to a 6-DOF manipulator. Although myCobot is operated with 3 Degree-Of-Freedom in this paper, it originally had 6 Degree-Of-Freedom, so it is desirable to implement methods for motion planning and path planning with 6 Degree-Of-Freedom.

References

[1] B. Siciliano, O. Khatib, Springer Handbook of Robotics, 2nd ed., Springer, 2016. doi:10.1007/978-3-319-32552-1.

[2] J.-C. Faugère, J.-P. Merlet, F. Rouillier, On solving the direct kinematics problem for parallel robots, Research Report RR-5923, INRIA, 2006. URL: https://hal.inria.fr/inria-00072366.

[3] C. M. Kalker-Kalkman, An implementation of Buchbergers' algorithm with applications to robotics, Mech. Mach. Theory 28 (1993) 523–537. doi:10.1016/0094-114X(93)90033-R.

[4] S. Ricardo Xavier da Silva, L. Schnitman, V. Cesca Filho, A Solution of the Inverse Kinematics Problem for a 7-Degrees-of-Freedom Serial Redundant Manipulator Using Gröbner Bases Theory, Mathematical Problems in Engineering 2021 (2021) 6680687. doi:10.1155/2021/6680687.

[5] T. Uchida, J. McPhee, Triangularizing kinematic constraint equations using Gröbner bases for real-time dynamic simulation, Multibody System Dynamics 25 (2011) 335–356. doi:10.1007/s11044-010-9241-8.

[6] T. Uchida, J. McPhee, Using Gröbner bases to generate efficient kinematic solutions for the dynamic simulation of multi-loop mechanisms, Mech. Mach. Theory 52 (2012) 144–157. doi:10.1016/j.mechmachtheory.2012.01.015.

[7] N. Horigome, A. Terui, M. Mikawa, A Design and an Implementation of an Inverse Kinematics Computation in Robotics Using Gröbner Bases, in: A. M. Bigatti, J. Carette, J. H. Davenport, M. Joswig, T. de Wolff (Eds.), Mathematical Software – ICMS 2020, Springer International Publishing, Cham, 2020, pp. 3–13. doi:10.1007/978-3-030-52200-1_1.

[8] S. Otaki, A. Terui, M. Mikawa, A Design and an Implementation of an Inverse Kinematics Computation in Robotics Using Real Quantifier Elimination based on Comprehensive Gröbner Systems, Preprint, 2021. doi:10.48550/arXiv.2111.00384, arXiv:2111.00384.

[9] M. Yoshizawa, A. Terui, M. Mikawa, Inverse Kinematics and Path Planning of Manipulator Using Real Quantifier Elimination Based on Comprehensive Gröbner Systems, in: Computer Algebra in Scientific Computing. CASC 2023, volume 14139 of *Lecture Notes in Computer Science*, Springer, 2023, pp. 393–419. doi:10.1007/978-3-031-41724-5_21.

[10] V. Weispfenning, Comprehensive Gröbner Bases, J. Symbolic Comput. 14 (1992) 1–29. doi:10.1016/0747-7171(92)90023-W.

[11] R. Fukasaku, H. Iwane, Y. Sato, Real Quantifier Elimination by Computation of Comprehensive Gröbner Systems, in: Proceedings of the 2015 ACM on International Symposium on Symbolic and Algebraic Computation, ISSAC '15, Association for Computing Machinery, New York, NY, USA, 2015, pp. 173–180. doi:10.1145/2755996.2756646.

[12] Elephant Robotics Co., Ltd., myCobot 280 M5, 2023. URL: https://www.elephantrobotics.com/mycobot-280-m5-2023, accessed 2024-05-04.

[13] G. Farin, Curves and Surfaces for CAGD: A Practical Guide, The Morgan Kaufmann Series in Computer Graphics, 5th ed., Morgan Kaufmann, 2002. doi:10.1016/B978-1-55860-737-8.X5000-5.

[14] K. M. Lynch, F. C. Park, Modern Robotics: Mechanics, Planning, and Control, Cambridge University Press, 2017.
[15] D. Kapur, Y. Sun, D. Wang, An efficient method for computing comprehensive Gröbner bases, J. Symbolic Comput 52 (2013) 124–142. doi:10.1016/j.jsc.2012.05.015.
[16] K. Nabeshima, CGS: a program for computing comprehensive Gröbner systems in a polynomial ring [computer software], 2018. URL: https://www.rs.tus.ac.jp/~nabeshima/softwares.html, accessed 2024-05-04.
[17] M. Noro, T. Takeshima, Risa/Asir — A Computer Algebra System, in: ISSAC '92: Papers from the International Symposium on Symbolic and Algebraic Computation, Association for Computing Machinery, New York, NY, USA, 1992, pp. 387–396. doi:10.1145/143242.143362.
[18] Wolfram Research, Inc., Mathematica, Version 13.1 [computer software], 2022. URL: https://www.wolfram.com/mathematica, accessed 2024-05-04.
[19] E. W. Dijkstra, A note on two problems in connexion with graphs, Numer. Math. 1 (1959) 269–271. doi:10.1007/BF01386390.
[20] simonritchie, Compute the shortest path of a graph using Dijkstra method and Python (in Japanese), 2023. URL: https://qiita.com/simonritchie/items/216eae753fc393da52af, accessed: 2024-05-04.
[21] R. Hatakeyama, A. Terui, M. Mikawa, Towards Trajectory Planning of a Robot Manipulator with Computer Algebra using Bézier Curves, in: SCSS 2024 Work-in-progress Proceedings, Open Publishing Association, 2024. To appear.
[22] S.-L. Guo, J. Duan, Y. Zhu, X.-C. Li, T.-W. Chen, Improved dijkstra algorithm based on fibonacci heap for solving the shortest path problem with specified nodes, in: Computer Science and Artificial Intelligence: Proceedings of the International Conference on Computer Science and Artificial Intelligence (CSAI2016), World Scientific, 2017, pp. 52–61. doi:10.1142/9789813220294_0008.