 DOI: 10.3217/978-3-85125-663-5-21

Computational Performance of the Forward and Inverse Kinematics of an Anthropomorphic Robot Arm

Christian HARTL-NESIC[1], Martin MEIRINGER[1]

I. INTRODUCTION AND MOTIVATION

In robotics, two fundamental coordinate spaces are utilized, i. e. the so called configuration space and the joint space. Most of the robot tasks are naturally defined in the configuration space (also known as task space) whilst actors and sensors of a robot operate in the joint space (e. g. encoders, electric motors, torque sensors ...). Therefore a bidirectional transformation is desired for describing a vector of one space in the other one.

Modern applications require increasing accuracy during task execution, which demands an increasing bandwidth of the control algorithms. To meet these increasing requirements, fast and efficient algorithms are necessitated for calculating the robot kinematics, i. e. the forward and inverse kinematics. While the forward kinematics describes the end effector pose (position and orientation) in the configuration space as a function of the joint space coordinates and the inverse kinematics describes the joint space coordinates as a function of the end effector pose. Two concepts are investigated in the course of this work: i) homogeneous coordinates, see, e. g., [3] and ii) dual quaternions, see, e. g., [1]. The calculations are performed for the 7-degree-of-freedom (DOF) anthropomorphic arm *KUKA LWR IV+*.

II. COMPUTATIONAL EFFORT FOR KINEMATICS

This work aims to compare the calculation costs of the robots kinematics between two well known methods of describing the kinematics. First, elemental kinematic operations are separately investigated, followed by the forward and inverse kinematics for the 7-DOF robot.

A. Method of Performance Calculation

The computational performance is quantified by the calculation costs, i. e. the number of additions, multiplications and function calls required to execute an algorithm and the mean time to numerically perform a calculation. All presented results, including the costs, where evaluated using the algebra system MAPLE 2018. The numerical results where calculated using MATLAB 2017b, using an office PC equipped with an Intel Core i7-6700K 4 GHz CPU and 16 GB RAM.

*This work was not supported by any organization.

[1]Faculty of Electrical Engineering, Institute for Automation and Control, TU Wien, Vienna, Austria {hartl, meiringer}@acin.tuwien.ac.at

B. Fundamental Transformations

The discrete transformation (translation and rotation) between two coordinate frames using homogeneous coordinates is given by the 4×4 matrix $\mathbf{T}$ given by

$$\mathbf{T} = \begin{bmatrix} 1 & \mathbf{0}^{\mathrm{T}} \\ \mathbf{d} & \mathbf{R} \end{bmatrix} , \tag{1}$$

where $\mathbf{d}$ represents the translation between the two frames and $\mathbf{R}$ denotes the rotation matrix representing the change in orientation. The same transformation, formulated in dual quaternions is given by

$$\underline{\mathbf{u}} = \mathbf{q}_R + \frac{1}{2}\varepsilon \mathbf{q}_R \otimes \mathbf{q}_d , \tag{2}$$

where $\mathbf{q}_R$ denotes the unit quaternion representing the change in orientation, $\mathbf{q}_d$ denotes the quaternion representing the translation between the two frames, ε is the dual unit and $\otimes$ denotes the quaternion multiplication (see [1]). The transformation matrix $\mathbf{T}$ in (1) and the transformation dual quaternion $\underline{\mathbf{u}}$ in (2) are used to perform discrete transformations, e. g. for changing the reference frame of a point in task space. By utilizing the well known Denavit-Hartenberg convention, see, e. g., [3], the relative transformations between two coordinate frames, attached to two consecutive links, can be directly parameterized. Comparing the computational costs of (1) and (2) gives a first idea concerning the computational performance of the two used methods. The costs are summarized in Table II. For each calculation, the costs without code optimization and the costs using the code optimization provided by MAPLE 2018 are given, where the non-optimized results are denoted with the ˜ symbol. Note, the transformations for this comparison where constructed using general, nonzero Denavit-Hartenberg parameters.

C. Forward Kinematics

The forward kinematics is given by the composition of N transformations, where N is the number of degrees-of-freedom. The forward kinematics for the end effector pose is then calculated by

$$\mathbf{f}_H(\theta) = \mathbf{T}_0^N(\theta) = \mathbf{T}_0^1(\theta_1)\mathbf{T}_1^2(\theta_2)\cdots\mathbf{T}_{N-1}^N(\theta_N) , \tag{3}$$

$$\mathbf{f}_{dQ}(\theta) = \underline{\mathbf{u}}_0^N(\theta) = \underline{\mathbf{u}}_0^1(\theta_1)\otimes\underline{\mathbf{u}}_1^2(\theta_2)\cdots\otimes\underline{\mathbf{u}}_{N-1}^N(\theta_N) , \tag{4}$$

where $\otimes$ denotes the dual quaternion multiplication (see [1]). The vector θ in (3) and (4) summarizes the relative joint angles of the robot[1]. The relative transformations $\mathbf{T}_{i-1}^i$ and $\underline{\mathbf{u}}_{i-1}^i$ are parameterized using the Denavit-Hartenberg

[1]The dependencies $\mathbf{T}(\theta)$ and $\underline{\mathbf{u}}(\theta)$ are omitted in the following for brevity.

i	a_i	α_i	d_i	θ_i
1	0	$\pi/2$	0	q_1
2	0	$\pi/2$	0	q_2
3	0	$\pi/2$	l_1	q_3
4	0	$\pi/2$	0	q_4
5	0	$\pi/2$	l_2	q_5
6	0	$\pi/2$	0	q_6
7	0	0	0	q_7

TABLE I

DENAVIT-HARTENBERG PARAMETERS FOR THE *KUKA LWR IV+*.

calculation	add	mult	fcn. calls	mean CPU time
$\tilde{\mathbf{T}}$ in (1)	2	6	14	-
$\mathbf{T}$ in (1)	2	6	4	-
$\underline{\tilde{\mathbf{u}}}$ in (2)	4	52	24	-
$\underline{\mathbf{u}}$ in (2)	4	20	4	-
$\tilde{\mathbf{f}}_H(\theta)$	88	238	327	4.109 µs
$\mathbf{f}_H(\theta)$	40	87	14	3.642 µs
$\tilde{\mathbf{f}}_{dQ}(\theta)$	312	1576	880	19.510 µs
$\mathbf{f}_{dQ}(\theta)$	38	88	14	4.398 µs
$\mathbf{f}_H^{-1}(\mathbf{T}^\star)$	1461	2570	52	86.138 µs
$\mathbf{f}_{dQ}^{-1}(\mathbf{T}^\star)$	1070	2258	184	173.104 µs

TABLE II

COST COMPARISON FOR FUNDAMENTAL TRANSFORMATIONS, FORWARD AND INVERSE KINEMATICS.

parameters given in Table I. The resulting calculation costs for the forward kinematics are summarized in Table II.

D. Inverse Kinematics

The inverse kinematics is a more challenging task and often not solvable analytically. Also the number of possible solutions can vary from zero to infinite solutions. For the inverse kinematics, the nonlinear equations (3) and (4) have to be solved for θ. This problem is formulated as

$$\theta = \mathbf{f}_H^{-1}(\mathbf{T}^\star) \,, \tag{5}$$

$$\theta = \mathbf{f}_{dQ}^{-1}(\underline{\mathbf{u}}^\star) \,, \tag{6}$$

where the superscript $\star$ denotes a given end-effector pose. The *KUKA LWR IV+* comprises a spherical sholder joint, an elbow joint and a spherical wrist, see, e. g., [3], this allows an analytical solution of the problems (5) and (6). Thanks to this construction, the kinematics can be separated in a position and orientation task, since the last three joints, the spherical wrist, only change the end effector orientation while leaving its position unchanged. Due to the redundancy of the considered robot, one joint angle remains an independent DOF. In this work, θ_3 is chosen as DOF. For the considered robot, the inverse kinematics problem results in eight sets of one-parametric solutions.

The inverse kinematics implementation, formulated using homogeneous coordinates, follows the work of Pfurner [2]. The method for calculating the inverse kinematics, formulated in dual quaternions is based on the arguments given by Pfurner. The different formulation of the kinematics changes the analytic structure of the obtained equations,

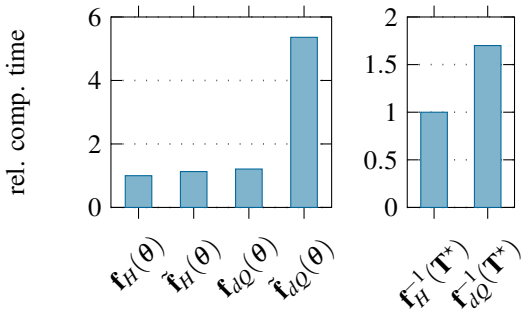

Fig. 1. Mean computation times for the forward and inverse kinematics.

but the dependencies of the equations remain unchanged. The resulting system of equations is solved using MAPLE 2018 following the procedure presented in [2]. The resulting calculation costs for the inverse kinematics are summarized in Table II. For practical applications, in addition to the calculation costs also the mean calculation time to execute an algorithm is of interest. These are also summarized in Table II and visualized in Figure 1. Solving the position part of the inverse kinematics problem, the dual quaternion formulation utilizes twice the analytical solution of a 4th degree polynomial. However, this yields multiple invalid solutions, which have to be eliminated using an additional equation. Thus, the homogeneous coordinate formulation is significantly faster.

III. CONCLUSION

In this work, the forward and inverse kinematics of a 7-DOF anthropomorphic arm, the *KUKA LWR IV+* were formulated using i) homogeneous coordinates and ii) dual quaternions and evaluated with respect to the computational costs and mean computation time.

By utilizing the code optimization functions provided by modern computer algebra systems, e. g. MAPLE 2018, the forward kinematics implementations perform almost equally, while the homogeneous coordinate formulation is slightly faster. The inverse kinematics implementations show a more significant difference in computation time, whereas the dual quaternion implementation is about 70 % slower. Hence, if the robot task does not prescribe the coordinate space, the presented results suggest to use homogeneous coordinates for calculating the robot kinematics.

ACKNOWLEDGMENT

This project was carried out within the framework of the course "Topics in Higher Geometry" held by Dr. techn. Georg Nawratil of the Institute of Discrete Mathematics and Geometry, TU Wien.

REFERENCES

[1] K. M. Lynch and F. C. Park, *Modern Robotics: Mechanics, Planning, and Control*. Cambridge University Press, July 2017.

[2] M. Pfurner, "Closed form inverse kinematics solution for a redundant anthropomorphic robot arm," *Computer Aided Geometric Design*, vol. 47, pp. 163–171, Oct. 2016.

[3] M. W. Spong, S. Hutchinson, and M. Vidyasagar, *Robot Modeling and Control*. Wiley New York, 2006, vol. 3.