Real-time Trajectory Generation for Both Arms of a Humanoid Robot

Chang-Young Jung and Jong-Hwan Kim

Department of Electrical Engineering, KAIST, 291 Daehak-ro, Yuseong-gu, Daejeon, 305-701, Republic of Korea
cyjung,johkim@rit.kaist.ac.kr

Abstract. This paper proposes a real-time trajectory generation algorithm for both arms of a humanoid robot. Since it is hard to find a closed form of inverse kinematics for each arm of seven degrees of freedom, the damped least-squares method is employed to obtain the inverse kinematics. The trajectory is generated by the minimum-jerk method to maximize the position accuracy. Considering the performance in computation time, a software SD/FAST is used to find a Jacobian matrix of the arm. Computer simulation was performed to verify the effectiveness of the proposed algorithm using a Webot simulator for the upper body of Mybot developed in the RIT Lab., KAIST. The results show that the proposed algorithm generates trajectory in real-time and it is robust to singularity.

Keywords: Humanoid robot, Robot arms trajectory generation, Damped least square, Minimum-jerk.

1 Introduction

Two-arm system of a humanoid robot consists of a trunk and two arms each of which can be regarded as an independent manipulator. Based on each manipulator control, the robot can manipulate both arms simultaneously and dexterously. Once the robot is given a task like pointing or reaching a position, it has to provide its arm's trajectory to reach the position. To make this possible, this paper proposes a novel algorithm that generates trajectories for both arms in real-time.

The research of manipulator has a long history. It first starts with the problem of inverse kinematics. As the number of robot's joint increases, nonlinear terms of inverse kinematics greatly increase. Therefore, it is impossible to find a closed form of inverse kinematics of the manipulator. Several methods have been introduced including neural network mapping [1], rapidly random tree in configuration space [2][3], and iterative method [4]. Even though the inverse kinematic equations are obtained, the generation of proper trajectory is still a difficult problem.

In this paper, an efficient real-time trajectory generation algorithm for both arms of humanoid robot is proposed by employing methods used in robot manipulator control. The damped least-squares method is used to obtain inverse kinematics of the arm of seven degrees of freedom. After finding the inverse kinematics, the minimum-jerk trajectory generation method is used to get the trajectory of the arm. Proposed algorithm is expected to be used as a core algorithm in task planning of both arms, where both arms need to be manipulated to do a commanded action. The effectiveness of the proposed algorithm is demonstrated through computer simulation using a Webot simulator for a model of Mybot, developed in the RIT Lab, KAIST.

This paper is organized as follows. Section 2 describes an arm of Mybot and its kinematics. Section 3 proposes a real-time trajectory generation algorithm for both arms of a humanoid robot. Section 4 presents simulations and the results for the Mybot arms. Concluding remarks along with future work follow in Section 5.

2 Kinematics of an Arm

SD/FAST which is a software developed by Symbolic Dynamics Inc. is used to obtain kinematic and dynamic equations of a humanoid arm [6]. This software uses Kane's formulation and symbolic manipulation to formulate the nonlinear equations of a multi-body mechanical system [5]. As each arm has seven degrees of freedom(**DOF**), the robot has redundant DOFs in 3D Cartesian space. The feature of left arm is presented in Fig. 1. Entire torso of Mybot consisting of two arms and trunk is also presented in Fig. 2.

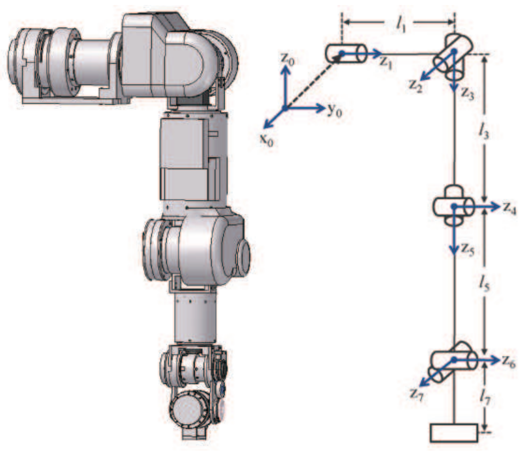


Fig. 1. Hardware Design of 7 DOF Mybot arm.

As shown in Fig. 2, two arms are located on the upper side of trunk. Therefore, there are three Cartesian coordinate systems in the right arm, left arm and trunk. The coordinate system of trunk is considered as a global or base coordinate system. Each arm has its own coordinate system of which the origin is

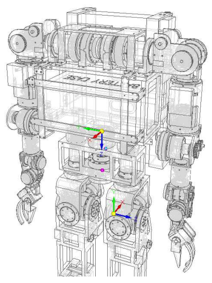


Fig. 2. Mybot's two arms and trunk.

located at the first joint of arm, and every coordinate system is easly derived only by multiplying translation transformation matrix to base frame. Therefore, in the rest of the paper, the trajectory generation method for only one arm is described. For the other arm, the same procedure is adopted. In other words, the same trajectory generation algorithm is applied for both arms with each arm's own coordinate system. The nominal values of left arm's physical feature used in SD/FAST are summarized in Table 1. '**SH**','**EL**', and '**WR**' represent shoulder, elbow, and wrist, respectively and '**X**','**Y**', and '**Z**' repectively mean the corresponding rotation axis with respect to the base frame.

Jacobian matrix $\mathbf{J}$, which transforms angular velocity into end-effector velocities in Cartesian space, is given as a 6×7 matrix as follow:

$$\begin{bmatrix} v_x \\ v_y \\ v_z \\ \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} = \mathbf{J}(\mathbf{q}) \begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \\ \dot{q}_3 \\ \dot{q}_4 \\ \dot{q}_5 \\ \dot{q}_6 \\ \dot{q}_7 \end{bmatrix} \tag{1}$$

where v and w denote linear and angular velocities of end-effector in the Cartesian coordinate system, respectively. The vector $\mathbf{q}$ represents joint angle, and $\dot{\mathbf{q}}$ indicates joint angular velocity. A velocity propagation method, which finds angular velocity of each joint from the base frame, is used to calculate the Jacobian matrix above.

3 Trajectory Generation Algorithm

The overall algorithm is shown in Algorithm 1. If the target position and target rotation are given to the robot, the algorithm generates tarajectory of an arm to reach the position. The following notations are used:

$$\mathbf{p}^0 = \begin{bmatrix} x^0 \; y^0 \; z^0 \; \psi^0 \; \theta^0 \; \phi^0 \end{bmatrix}^T \quad , \quad \mathbf{p}^N = \begin{bmatrix} x^N \; y^N \; z^N \; \psi^N \; \theta^N \; \phi^N \end{bmatrix}^T \tag{2}$$

Table 1. Nominal values of dynamic parameter of 7 DOF humanoid arm

Name	**Link**	**Length** (m)	**Mass** (kg)	**Inertia** ($kg \cdot m^2$)
SHY	1	0.2 (0.1)	1.0	$\begin{bmatrix} 0.1 & 0.01 & 0.01 \\ 0.01 & 0.05 & 0.01 \\ 0.01 & 0.01 & 0.1 \end{bmatrix}$
SHX	2	0.0	0.8	$\begin{bmatrix} 0.03 & 0.005 & 0.005 \\ 0.005 & 0.08 & 0.005 \\ 0.005 & 0.005 & 0.08 \end{bmatrix}$
SHZ	3	0.2 (0.1)	1.0	$\begin{bmatrix} 0.15 & 0.01 & 0.01 \\ 0.01 & 0.15 & 0.01 \\ 0.01 & 0.01 & 0.05 \end{bmatrix}$
ELY	4	0.0	0.5	$\begin{bmatrix} 0.05 & 0.005 & 0.005 \\ 0.005 & 0.01 & 0.005 \\ 0.005 & 0.005 & 0.05 \end{bmatrix}$
WRZ	5	0.2 (0.1)	0.5	$\begin{bmatrix} 0.04 & 0.004 & 0.004 \\ 0.004 & 0.04 & 0.004 \\ 0.004 & 0.004 & 0.008 \end{bmatrix}$
WRY	6	0.0	0.3	$\begin{bmatrix} 0.03 & 0.003 & 0.003 \\ 0.003 & 0.006 & 0.003 \\ 0.003 & 0.003 & 0.03 \end{bmatrix}$
WRX	7	0.05 (0.05)	0.2	$\begin{bmatrix} 0.006 & 0.003 & 0.003 \\ 0.003 & 0.03 & 0.003 \\ 0.003 & 0.003 & 0.03 \end{bmatrix}$

where $\mathbf{p}^0$ and $\mathbf{p}^N$ represent initial pose and final pose of robot arm, respectively, and $[x, y, z]^T$ and $[\psi, \theta, \phi]^T$ represent the position and orientation, respectively. As the number of samples, $\mathbf{N}$ increases, more interpolation points are generated in trajectory. After interpolated joint values $\mathbf{q}^i_{ref}$ are obtained, the robot follows the trajectory by assigning joint values.

This algorithm is applied to both arms identically. As mentioned earlier, since each arm has its own coordinate system, the algorithm is seperately adopted by each arm based on its own coordinate system. Once the target position is given based on global coordinate system, the position value is converted into each arm's coordinate system and the algorithm is carried out.

Proposed algorithm employs the damped least-squares method [7] in order to calculate inverse kinematics. Since robot arm has 7 degrees of freedom with redundant joints, it is hard to find a closed form of inverse kinematics. The damped least-squares method, also called as Levenberg-Marquardt method, finds inverse kinematics by using linear velocity, angular velocity and Jacobian matrix iteratively. The obtained solution is numerically stable [8]. The cost function of the damped least-squares method is defined as follows [5]:

$$L(\mathbf{p},\mathbf{q})=\left({}^{j+1}\mathbf{p}-\mathbf{p}^i\right)^T r_1\left({}^{j+1}\mathbf{p}-\mathbf{p}^i\right)+\left({}^{j+1}\mathbf{p}-{}^{j}\mathbf{p}\right)^T r_2\left({}^{j+1}\mathbf{p}-{}^{j}\mathbf{p}\right)+\left({}^{j+1}\mathbf{q}-{}^{j}\mathbf{q}\right)^T r_3\left({}^{j+1}\mathbf{q}-{}^{j}\mathbf{q}\right) \quad (3)$$

Algorithm 1 Trajectory Generation Algorithm

```
 1: procedure DAMPED LEAST-SQUARES($\mathbf{p}$)                ▷ generating $\mathbf{q}$ from $\mathbf{p}$
 2:     $\mathbf{p}^0, \mathbf{p}^N \leftarrow \mathbf{p}$                          ▷ $N$: the number of samples
 3:     while $error_{norm} < \mathbf{e}$ do          ▷ $error_{norm} = \mathbf{p}_{desired} - \mathbf{p}_{current}$
 4:         $\Delta\mathbf{q} \leftarrow L(\mathbf{p}, \mathbf{q})$           ▷ $L(\mathbf{p}, \mathbf{q})$ : Levenberg-Marquardt method
 5:         $\mathbf{q} = \mathbf{q} + \Delta\mathbf{q}$
 6:         $\mathbf{p}_{current} \leftarrow FK(\mathbf{q})$                  ▷ $FK$ : Forward Kinematics
 7:     end while
 8:     return $\mathbf{q}^0_{ref}, \mathbf{q}^N_{ref}$
 9: end procedure

10: procedure MINIMUM-JERK TRAJECTORY($\mathbf{q}^0_{ref}, \mathbf{q}^N_{ref}$)
11:     $\mathbf{q}^i_{ref}, \dot{\mathbf{q}}^i_{ref}, \ddot{\mathbf{q}}^i_{ref} \leftarrow \mathbf{q}_{ref}$
12:     return $\mathbf{q}^i_{ref}$                                  ▷ $i$ : N samples
13: end procedure
```

where j is the iteration number, r_1, r_2 and r_3 are constants. When partial derivative according to ${}^{j+1}\mathbf{q}$ of $\mathbf{L}$ is zero, the joint value error becomes a minimum. Therfore, after taking partial derivative and rearranging the formula, $\Delta^j\mathbf{q}$ can be expressed as follows:

$$\Delta^j\mathbf{q} = \mathbf{J}^T \left(\mathbf{J}\mathbf{J}^T + \lambda^2\mathbf{I}\right)^{-1} \Delta\mathbf{p} \tag{4}$$

where r_1 and r_2 is set to 1 and r_3 is set to λ^2. Accuracy and feasibility of the result are determined by the damping factor λ. The detailed formulations can be obatined from [5]. In order to solve the above equation, matrix inversion calculation is required. It is achieved by using a LDLT decomposition [5].

After calculating inverse kinematics, the outputs obtained by the damped least-squares are only initial and final joint values of required pose. The entire trajectory, which connects between start position and end position, should be generated. In the view of joint space, joint values between initial and final joint values should be determined by the interpolation method. There are many methods that generate a trajectory between two points. The following minimum-jerk trajectory method is used to maximize the accuracy of joint position:

$$\begin{aligned}
\mathbf{q}^i_{ref} &= \mathbf{q}^0 + \left(\mathbf{q}^N - \mathbf{q}^0\right)(10k_3 - 15k_4 + 6k_5) \\
\dot{\mathbf{q}}^i_{ref} &= \frac{\left(\mathbf{q}^N - \mathbf{q}^0\right)}{N}(30k_2 - 60k_3 + 30k_4) \\
\ddot{\mathbf{q}}^i_{ref} &= \frac{\left(\mathbf{q}^N - \mathbf{q}^0\right)}{N^2}(60k_1 - 180k_2 + 120k_3)
\end{aligned} \tag{5}$$

with

$$k_1 = \frac{i}{N},\ k_2 = k_1^2,\ k_3 = k_1^3,\ k_4 = k_1^4,\ k_5 = k_1^5$$

where i is the sample point index. As the jerk is a rate of acceleration, the position error is increased as the jerk increases. Furthermore, minimum-jerk trajectory not only minimizes the position error, but also it makes the trajectory similar to human arm trajectory [9].

4 Computer Simulations

Webots simulator, which is a development environment to model, program and simulate mobile robots, was used to test the proposed algorithm. The 3D model and its nominal values are given in Fig. 2 and Table 1. The algorithm was implemented by C++, and the simulation was performed on a desktop computer. In the given experimental environment, real world's physics were not considered in order to concentrate on testing the feasibility of generating trajectory to the given position. The target position set $\mathbf{q}^N$ could be obtained from user in various ways. In this experiment, the target position sets in Fig. 3 were applied to the robot to measure the time for generating trajectories for two arms. The accuracy was determined by analyzing error norm, $\sum_i |p_{i,desired} - p_{i,current}|$.

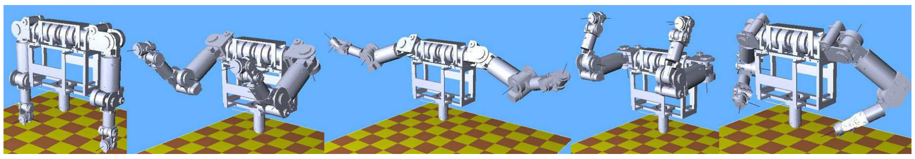
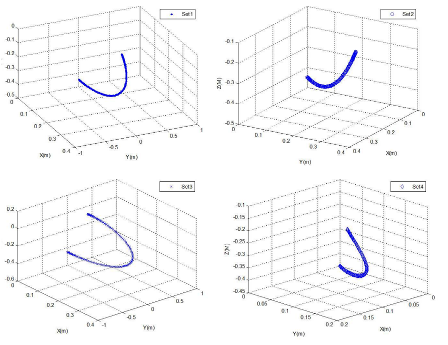


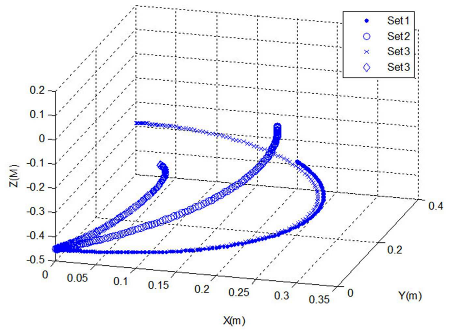
Fig. 3. Test position sets.

$$\begin{aligned} StartPosition : \mathbf{p}^0 &= \begin{bmatrix} 0.0 & 0.0 & -0.45 & 0.0 & 0.0 & 0.0 \end{bmatrix}^T \\ Position1 : \mathbf{p}^1 &= \begin{bmatrix} 0.3 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \end{bmatrix}^T \\ Position2 : \mathbf{p}^2 &= \begin{bmatrix} 0.2 & 0.3 & -0.15 & 0.0 & 0.0 & 0.0 \end{bmatrix}^T \\ Position3 : \mathbf{p}^3 &= \begin{bmatrix} 0.1 & 0.0 & 0.1 & 0.0 & 0.0 & 0.0 \end{bmatrix}^T \\ Position4 : \mathbf{p}^4 &= \begin{bmatrix} 0.1 & 0.12 & -0.18 & 0.0 & 0.0 & 0.0 \end{bmatrix}^T \end{aligned} \tag{6}$$

For given target position sets, the experimental results are summarized in Table 2 and Fig. 4. The trajectories in Fig. 4 present left arm end-effector's trajectory in Cartesian coodrdinate system. Fig. 4 (a) shows the trajectory of each position set (6), and all of them are drawn in one graph in Fig. 4 (b).
Every position sets start from leftmost pose in Fig. 3. Smooth trajectories are generated as shown in Fig. 4. Both horizontal and vertical motion trajectories were generated successfully.



(a)



(b)

Fig. 4. End-effector trajectories for test sets

Table 2. Damped least-squares iteration number and computation time

Set	iteration	computation time (ms)	error norm (m)
Position set 1	2612	37.0542	0.000001
Position set 2	2586	50.1257	0.000001
Position set 3	2568	35.9959	0.000001
Position set 4	2518	54.0743	0.000001

The computation time can be varied according to the computation power. In this simulation, the computation time took more than 30ms but below 60ms. However, trajectory generation is not required to be performed in every control period which is usually less than 30ms. While following a current trajectory, the robot can generate the next trajectory in 60ms for the next motion. Modern robot real-time system is required under 1ms on motion trajectory generation. However, due to iterative way of finding a inverse kinematics, it is hard to reduce computation time by using this generation algorithm. Though it has a limitation on trajectory generation time for fast control system, the robot can make a path by itself not just only follwoing pre-programmed path according to given command. Therefore, proposed algorithm can be applied to the real-time trajectory generation of both arms of a humanoid robot.

5 Conclusions

This paper proposed a real-time trajectory generation algorithm for both arms of a humanoid robot. Once the target position in Cartesian coodrinate system was given, the damped least-squares method was used to find inverse kinematics of start and final positions. After obtaining the inverse kinematics, the algorithm generated an interpolated trajectories which are minimum-jerk trajectories. Computer simulation demonstrated its effectiveness successfully. That is, the proposed trajectory generation algorithm shows real-time ability and robustness to singularity. The proposed algorithm is expected to be utilized in robot motion and task planning applications.

For the future work, robot's dynamics and real world physics should be considered to be applied to a real robot. The robot should control its arm by torque based on generated trajectory. In addition, the algorithm still makes a trajectory without considering self body collision. In other words, the algorithm generates a trajectory which crosses the trunk or the other hand for reaching a certain position. The algorithm has to make an trajectory with self body collision avoidance as well as external obstacles.

Acknowledgements

This research was supported by the MKE (The Ministry of Knowledge Economy), Korea, under the Human Resources Development Program for Convergence Robot Specialists support program supervised by the NIPA (National IT Industry Promotion Agency) (NIPA-2012-H1502-12-1002)

References

1. Nikolas J. Hemion, Frank Joublin, Katharina J. Rohlfing (2012) Integration of Sensorimotor Mappings by Making Use of Redundancies, Paper presented at the IEEE World Congress on Computational Intelligence, Brisbane, Australia, 10-15 June 2012
2. Dominik Bertram, James Kuffner, Ruediger Dillmann, Tamim Asfour (2006) An Integrated Approach to Inverse Kinematics and Path Planning for Redundant Manipulators, Paper presented at the IEEE International Coference of Robotics and Automation, Orlando, Florida, May 2006
3. Mike Vande Weghe, Dave Ferguson, Siddhartha S. Srinivasa (2007) Randomized Path Planning for Redundant Manipulator without Inverse Kinematics, Paper presented at the IEEE International Conference of Robotics and Automation Society, Roma, Italy, Nov. 2007
4. B. Subudhi, A. S. Morris (2009) Soft Computing methods applied to the control of a flexible robot manipulator, Applied Soft Computing 9:149-158
5. In-Won Park (2012) Kinematic Calibration, Optimal Trajectory Generation, and MOEA-based Optimal Posture Control of Humanoid Robot. Dissertation, KAIST
6. M. Hollars, D. Rosenthal and M. (1991) Sherman SD/FAST. Symbolic Dynamics, Inc.
7. Y. Nakamura and H. Hanafusa (1986) Inverse kinematic solutions with singularity robustness for robot manipulator control, ASME J. Dyn. Syst. Meas. Control, 108:163-171
8. C. Wampler (1986) Manipulator inverse kinematic solutions based on vector formulation and damped least-squares methods, IEEE Trans. Syst, Man Cyber., 16:93-101
9. T. Flash and N. Hogan (1985) The coordination of arm movements: an experimentally confirmed mathematical model, J. Neuroscience, 3:1688-1703