Robust Modular Inverse Kinematics for Gesture Imitation in an Upper-Body Humanoid Robot

Keng Peng Tee, Rui Yan, Yuanwei Chua, Haizhou Li and Zhiyong Huang

***Abstract*— In this paper, we present a method of computing the joint angles for an upper body humanoid robot corresponding to task space motion data from a human demonstrator. Using a divide-and-conquer approach, we group the motors into pan-tilt and spherical units, and solve the inverse kinematics in a modular fashion based on the derivative of the inverse tangent function of the relevant task space variables. For robustness to kinematic singularity, we add a regularization parameter that vanishes whenever the task variables are outside a neighborhood of zero. Furthermore, we perform scaling of the joint angles and velocities to ensure that their limits are satisfied and smoothness preserved. Simulation study on a 7 degree-of-freedom (DOF) robot arm shows a tradeoff of tracking accuracy in a neighborhood of each singularity in favor of robustness, but high accuracy is recovered outside this neighborhood. Experimental implementation of the proposed inverse kinematics on a 17-DOF upper-body humanoid robot shows that user-demonstrated gestures are well-replicated by the robot.**

I. Introduction

In recent years, there have been significant efforts to enrich human-robot interactions, and to make robots more appealing, by, for example, performing human-like gestures in accompaniment to speech. A promising paradigm for robots to learn new gestures is by imitation, wherein the robot directly mimics the gestures of the human demonstrator [1], [2], [3]. By providing a natural and intuitive interface for teaching a robot how to perform coordinated gestures involving many degrees of freedom, robotic motion programming can be made accessible to common users unskilled in robotics.

Gesture imitation involves converting a perception of the demonstrated gesture into a sequence of robot motor responses so as to replicate the gesture. Depending on the nature of the movements, we may be interested in different task features to imitate. For example, when imitating object interactions, the actual position of the end-effector is important. However, for gestures, the actual positions of the end-effectors are arguably not as important as the orientation of a body segment with respect to an adjacent one. Hence, our approach to gesture imitation is to imitate the relative orientations of the body segments of the human demonstrator. Essentially, this can be cast as an inverse kinematics problem where a *non-redundant* set of joint space angles that produce the targeted task space configuration is to be computed.

This work is supported by the A*STAR SERC Inter-RI Robotics Programme.

The authors are with the Institute for Infocomm Research, A*STAR, Singapore 138632
`{kptee,ryan,ychua,hli,zyhuang}@i2r.a-star.edu.sg`

For solving the inverse kinematics problem, much attention has been devoted to local solutions that are velocity based, since global solutions such as the inverse transform method [4], typically have difficulties in obtaining an appropriate solution from all possible solutions [5]. This problem has been alleviated with the use of nonlinear optimization techniques [6], [7], as well as pre-classification and parametrization of the solution manifolds [8]. Another problem with global solutions associated with the use of inverse tangent operation $\operatorname{atan2}$ is that there may be discontinuous jumps when about $\pm 180^\circ$, since the range is only defined in the interval $(-180^\circ, 180^\circ]$.

On the other hand, velocity based methods typically require an approximate inversion of the Jacobian, and include pseudoinverse methods [9] and Jacobian transpose methods [10], [11], among others. To arrest drift inherent in open loop integration of velocity, closed loop inverse kinematics schemes have been employed [10]-[13]. For robustness against kinematic singularities where the Jacobian loses rank, the method of damped least squares, based on a constant regularization constant, has been proposed [14], [15]. However, poor choice of the regularization constant can lead to inaccurate solutions even in regions free of singularities. To circumvent this problem, *selectively* damped least squares, which rely on singular value decomposition (SVD) of the Jacobian to vary the damping parameter, has been studied in [16], [17], [18]. The drawback is that SVD computations can be slow [18], especially when dealing with large number of degrees of freedom, and thus may not be suitable for online interactive applications such as gesture imitation.

To deal with kinematic constraints such as joint and velocity limits, various approaches have been used. An algorithm to ensure joint limit satisfaction is proposed in [1] and involves scaling the joint trajectory non-uniformly around a neighborhood of the segment transgressing the joint limits while retaining most of the relevant motion that lie within the limits. A different approach based on the weighted least norm is proposed in [19], [20] such that joint movement is mediated by a weight value that varies from unity at the middle of the joint limit range to zero at the joint limits. To satisfy velocity limits, the method presented in [1] averages the results of simulated velocity-limited tracking controller run forward and backward in time, while a different approach in [19] adapts the time steps at which the joint velocity violates the limits. While these methods of joint and velocity scaling ensure continuity of signal across time periods with different scaling factors, they do not ensure smoothness, and may lead to undesirable abrupt jumps in the derivative signal at those transition times.

In this paper, we propose a method of inverse kinematics for an upper body humanoid robot performing gesture imitation, based on a divide-and-conquer approach where the motors are grouped into pan-tilt and spherical modules, as described in Section II. For each module, we design adaptation laws for the joint angles by differentiating an inverse tangent function of the relevant task space variables obtained via consistent transformation matrices, and incorporating feedback error to mitigate drift. To provide robustness to kinematic singularity, at which the denominator of the adaptation law becomes zero, we add, to the denominator, a regularization parameter that is activated only when in a neighborhood of zero. Under the proposed modular framework, adapting this regularization parameter is simple and fast to compute, and do not rely on computationally expensive SVD operations. We account for kinematic constraints of the robot by employing scaling algorithms that ensure smoothness and avoid abrupt jumps in the derivative signal at all times, as shown in Section IV. Results of simulations and experiments are shown in Sections III and V, including a numerical comparison study of the proposed method against a Jacobian based method, as well as an implementation on a 17-DOF upper-body humanoid robot for online imitation of a human demonstrator's gestures.

II. Robust Modular Inverse Kinematics

Consider an upper body humanoid robot with kinematic configuration as shown in Figure 1. For analytical purpose, the configuration can be decomposed into two types of joint modules, namely 5 pan-tilt modules for the torso and arms, and 3 spherical modules for the head and wrists. Grouping them this way allows us to formulate and solve the inverse kinematics problem for gesture imitation in a modular manner. Since our approach to gesture imitation is to imitate the relative orientations of the body segments of the human demonstrator, dealing with redundancy is not required.

Coordinate frames are fixed to each link of the robot, and the zero configuration (i.e. all joint angles zero) is defined as in Figure 1A. The orientation of the coordinate frames are fixed according to the orientation of the modular units, such that the notation used in the determination of the joint angles for each module is consistent. In this way, the same algorithms for computing the angles for one joint module can be conveniently re-used for another joint module. Figure 1A shows that from the neck to each of the collars, the coordinate frame is rotated about the x-axis by $90°$. From the each collar to the corresponding upperarm, the coordinate frame is rotated about the y-axis by $90°$.

Our approach to gesture imitation is to imitate the relative orientations of the body segments of the human demonstrator, which means that the task space positions of all joints (waist, neck, shoulders, elbows, and wrists), as well as orientations of the end-effectors (hands and head), are used in the computation of the corresponding robot joint angles.

Let q be the robot joint angles, and $X = \{X_{pos}, X_{orient}\}$ is a 51-dimensional vector containing the task space positions of the robot joints $X_{pos} = \{X_{pos,j}\}$, $j \in \{1, 2H, 2L, 3L, 4L, 2R, 3R, 4R\}$, as well as the orientation of the end-effectors $X_{orient} = \{X_{orient,j}\}$, $j \in \{2H, 4L, 4R\}$. For ease of convenience, each 3×3 orientation matrix is written in a 9×1 vector form $X_{orient,j}$ by concatenating the columns consecutively. Then, the forward kinematics is represented by

$$X = f(q) \tag{1}$$

where $f(\cdot)$ is a nonlinear mapping. Then, the inverse kinematics problem is to solve for

$$q = f^{-1}(X) \tag{2}$$

The first requirement is to solve the inverse kinematics in a smooth and robust manner. Not only is the mapping from task space to joint space nonlinear and one-to-many, but it is also ill-conditioned when the robot is near its singular configurations. Also, the complexity increases with the number of DOFs of the robot. If the inverse kinematics is not robustly solved, then the robot may exhibit rapid and spurious changes in configuration. The second requirement is to re-plan the motion to ensure that the joint and velocity limits are satisfied simultaneously.

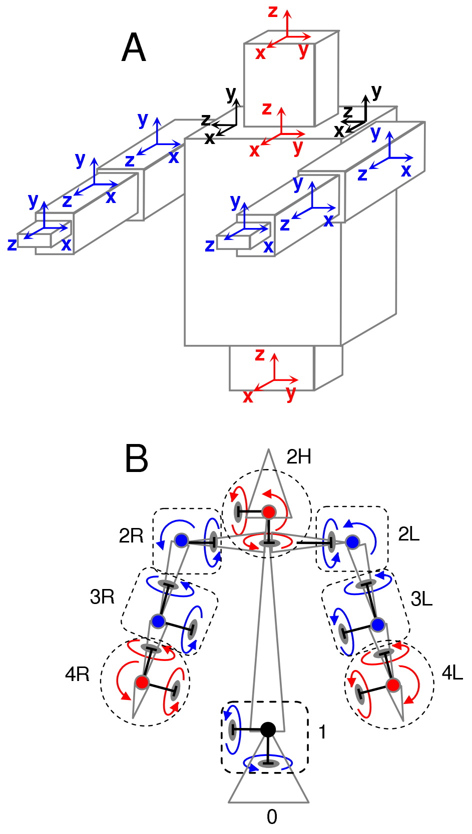


Fig. 1. (A) Zero configuration of the upper body humanoid robot, where coordinate frame orientation depends on body part. (B) The robot degrees-of-freedom are grouped into 2-DOF pan-tilt modules (squares) and 3-DOF spherical modules (circles).

A. 2-DOF Pan-Tilt Module

For each pan-tilt unit, as illustrated in Figure 2, let (x, y, z) be the task space information, specifically the endpoint position of the link with respect to a local coordinate frame (O_x, O_y, O_z) at the base of the pan-tilt joint.

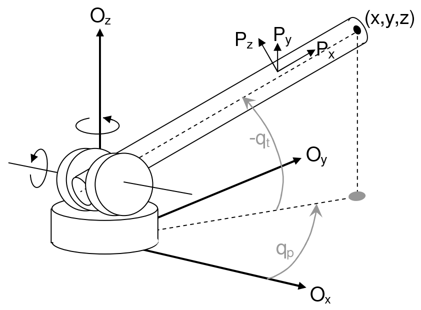


Fig. 2. Pan-tilt module.

It appears straightforward to compute, via geometrical relationship, the pan angle q_p and the tilt angle q_t as follows:

$$\begin{aligned} q_p &= \tan^{-1}\left(\frac{y}{x}\right) \\ q_t &= \tan^{-1}\left(\frac{x_r}{z}\right) \end{aligned} \qquad (3)$$

where

$$x_r = x\cos q_p + y \sin q_p \qquad (4)$$

is the x-coordinate of the new frame after the original frame is rotated by q_p about the z-axis. However, due to the fact that $\mathrm{atan2}(\cdot) \in (-180°, 180°]$, there may be discontinuous jumps in the solution (3) when crossing $\pm 180°$.

For smooth motion, we adopt a derivative-based approach to compute the angles. Taking the derivative of (3) yields

$$\dot{q}_p = \frac{x\dot{y} - y\dot{x}}{x^2 + y^2} \qquad (5)$$

$$\dot{q}_t = \frac{z\dot{x}_r - x_r\dot{z}}{x_r^2 + z^2} \qquad (6)$$

and we simply integrate the above quantities over time to obtain $q_p(t)$ and $q_t(t)$. At this point, it is clear that two problems exist, namely drift and singularity. Drift results from numerical integration wherein errors accumulate over time, while singularity occurs when $x^2 + y^2 = 0$.

To mitigate drift, we augment a feedback term to the cartesian velocity based on the error in the estimated cartesian position. In other words, we replace $\dot{x}$ by $(\dot{x} - k(\hat{x} - x))$, $\dot{y}$ by $(\dot{y} - k(\hat{y} - y))$, $\dot{z}$ by $(\dot{z} - k(\hat{z} - z))$, and $\dot{x}_r$ by $(\dot{x}_r - k(\hat{x}_r - x_r))$, where k is a positive constant. The position estimates are

$$\begin{aligned} \begin{bmatrix} \hat{x} \\ \hat{y} \\ \hat{z} \end{bmatrix} &= \mathrm{rot_z}(q_p)\mathrm{rot_y}(q_t)P^T \begin{bmatrix} L \\ 0 \\ 0 \end{bmatrix} \\ \hat{x}_r &= \hat{x}\cos q_p + \hat{y}\sin q_p \end{aligned} \qquad (7)$$

where $L = \sqrt{x^2 + y^2 + z^2}$ is the position magnitude, $\mathrm{rot_z}, \mathrm{rot_y} \in \mathcal{SO}(3)$ are elementary rotation matrices:

$$\begin{aligned} \mathrm{rot_z}(\star) &= \begin{bmatrix} \cos(\star) & -\sin(\star) & 0 \\ \sin(\star) & \cos(\star) & 0 \\ 0 & 0 & 1 \end{bmatrix} \\ \mathrm{rot_y}(\star) &= \begin{bmatrix} \cos(\star) & 0 & \sin(\star) \\ 0 & 1 & 0 \\ -\sin(\star) & 0 & \cos(\star) \end{bmatrix} \end{aligned} \qquad (8)$$

and $P \in \mathcal{SO}(3)$ depends on the location of the pan-tilt module (see Figure 1A):

$$P = \begin{cases} \mathrm{rot_x}(90°) & \text{if shoulder pan-tilt} \\ \mathrm{rot_y}(90°) & \text{if elbow pan-tilt} \\ I_{3\times 3} & \text{if waist pan-tilt} \end{cases} \qquad (9)$$

For robustness against singularity, we add to the denominator of (5) a *time-varying* regularization parameter $\varepsilon(x(t), y(t))$ as follows:

$$\varepsilon(x,y) = \begin{cases} 0, & x^2 + y^2 > \beta_2 \\ \varepsilon_0, & x^2 + y^2 < \beta_1 \\ \varepsilon_0 + a_1(x^2 + y^2 - \beta_1)^3 \\ +a_2(x^2 + y^2 - \beta_1)^2, & \beta_1 \le x^2 + y^2 \le \beta_2 \end{cases} \qquad (10)$$

where β_1, β_2, ε_0 are small positive constants, and

$$\begin{aligned} a_1 &= 2\varepsilon_0/(\beta_2 - \beta_1)^3 \\ a_2 &= -3\varepsilon_0/(\beta_2 - \beta_1)^2 \end{aligned} \qquad (11)$$

are obtained by considering boundary constraints when fitting a third order polynomial between the two extrema of ε. A schematic illustration of ε is shown in Figure 3. The regularization parameter adapts according to position (x, y) and is only active when (x, y) is in a small neighborhood of zero. In this way, the compromise of accuracy of tracking in favor of robustness to singularity is localized to a small region, and accurate tracking is recovered in the rest of the space.

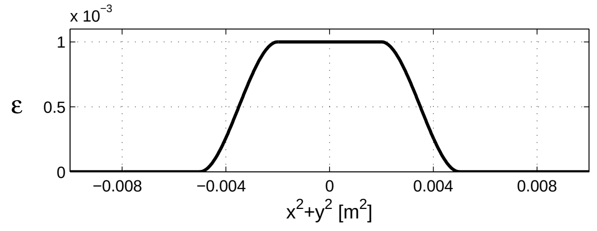


Fig. 3. The regularization parameter ε is only active in a small neighborhood of $(x, y) = 0$, but zero otherwise.

This leads to the following adaptation law for the pan and tilt angles:

$$\dot{q}_p = \frac{x(\dot{y} - ke_y) - y(\dot{x} - ke_x)}{x^2 + y^2 + \varepsilon(x,y)} \qquad (12)$$

$$\dot{q}_t = \frac{z(\dot{x}_r - ke_{x_r}) - x_r(\dot{z} - ke_z)}{x_r^2 + z^2} \qquad (13)$$

where $e_{(\bullet)} := \hat{(\bullet)} - (\bullet)$. The solutions $q_p(t)$ and $q_t(t)$ are initialized according to (3) by way of the $\mathrm{atan2}(\cdot)$ function.

Note that the residue $\varepsilon(x, y)$ only appears in (12), not (13), due to the fact that $x_r^2 + z^2 = L^2 \neq 0$. After solving for q_p and q_t, the orientation of the link with respect to the base frame is given by

$$R_p = \mathrm{rot_z}(q_p)\mathrm{rot_y}(q_t) \tag{14}$$

B. 3-DOF Spherical Module

Spherical units comprising 3 axes of rotation occur at the end-effectors, namely the two hands and the head. The orientation matrix of the hand with respect to the forearm can be decomposed into 3 ordered rotations $\mathrm{rot_z}, \mathrm{rot_y}, \mathrm{rot_x}$ about the local axes z, y, x respectively:

$$\begin{aligned} R_s &= \{r_{ij}\}, \quad i, j = 1, 2, 3 \\ &= \mathrm{rot_z}(q_1)\mathrm{rot_y}(q_2)\mathrm{rot_x}(q_3) \end{aligned} \tag{15}$$

where $\mathrm{rot_z}$ and $\mathrm{rot_y}$ are defined in (8), and

$$\mathrm{rot_x}(\star) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\star) & -\sin(\star) \\ 0 & \sin(\star) & \cos(\star) \end{bmatrix} \tag{16}$$

By right-multiplying both sides of (15) with the transpose of $\mathrm{rot_z}(q_3)$, we obtain

$$R_s \ \mathrm{rot_x}^T(q_3) = \mathrm{rot_z}(q_1)\mathrm{rot_y}(q_2) \tag{17}$$

Equating matrix coefficients on both sides of the above equation yields

$$\begin{aligned} q_3 &= \tan^{-1}\left(\frac{r_{32}}{r_{33}}\right) \\ q_2 &= \tan^{-1}\left(\frac{-r_{31}}{r_{32}\sin q_3 + r_{33}\cos q_3}\right) \\ q_1 &= \tan^{-1}\left(\frac{r_{13}\sin q_3 - r_{12}\cos q_3}{r_{22}\cos q_3 - r_{23}\sin q_3}\right) \end{aligned} \tag{18}$$

Similar to our approach for a pan-tilt unit, as described in Section II-A, we take the derivatives of the right hand sides of (19), and augment feedback terms and a regularization parameter. This leads to the adaptation laws for the joint angles

$$\begin{aligned} \dot{q}_3 &= \frac{r_{33}(\dot{r}_{32} - ke_{r_{32}}) - r_{32}(\dot{r}_{33} - ke_{r_{33}})}{r_{32}^2 + r_{33}^2 + \varepsilon(r_{32}, r_{33})} \\ \dot{q}_2 &= \frac{r_{31}(\dot{\alpha}_1 - ke_{\alpha_1}) - \alpha_1(\dot{r}_{31} - ke_{r_{31}})}{r_{31}^2 + \alpha_1^2} \\ \dot{q}_1 &= \frac{\alpha_3(\dot{\alpha}_2 - ke_{\alpha_2}) - \alpha_2(\dot{\alpha}_3 - ke_{\alpha_3})}{\alpha_2^2 + \alpha_3^2} \end{aligned} \tag{19}$$

where the regularization parameter $\varepsilon(\cdot, \cdot)$ is defined in (10), and

$$\begin{aligned} \alpha_1 &:= r_{32}\sin q_3 + r_{33}\cos q_3 \\ \alpha_2 &:= r_{13}\sin q_3 - r_{12}\cos q_3 \\ \alpha_3 &:= r_{22}\cos q_3 - r_{23}\sin q_3 \end{aligned} \tag{20}$$

Note, from (19), that only the denominator of the right hand side of the $\dot{q}_3$ equation contains ε. This is due to the fact that both $(r_{31}^2 + \alpha_1^2)$ and $(\alpha_2^2 + \alpha_3^2)$ are bounded away from zero.

The orientation estimates used in the feedback error terms are given by

$$\begin{aligned} \hat{r}_{31} &= -\sin q_2 \\ \hat{r}_{32} &= \cos q_2 \sin q_3 \\ \hat{r}_{33} &= \cos q_2 \cos q_3 \\ \hat{\alpha}_1 &= \cos q_2 \\ \hat{\alpha}_2 &= \sin q_1 \\ \hat{\alpha}_3 &= \cos q_1 \end{aligned} \tag{21}$$

These are obtained from (15) and (17) by comparing matrix coefficients on both sides of the equations. The solutions $q_1(t)$, $q_2(t)$ and $q_3(t)$ are initialized according to (18) by way of the $\mathrm{atan2}(\cdot)$ function.

C. Module-to-Module Transformation Matrices

The pan-tilt module adaptation law in (12)-(13) only derives pan and tilt angles in a local coordinate frame for a single pan-tilt unit. To ensure that the same equations can be consistently used for adjacent pan-tilt units in a recursive, modular fashion, an additional coordinate transformation matrix:

$$P_j = \begin{cases} \mathrm{rot_x}(90^\circ) & \text{if } j \in \{2L, 2R\} \\ \mathrm{rot_y}(90^\circ) & \text{if } j \in \{3L, 3R\} \\ I & \text{if } j \in \{1, 2H, 4L, 4R\} \end{cases} \tag{22}$$

is required when computing joint angles for some pan-tilt modules.

As shown in Figure 1A, there are orientation offsets at the zero configuration, specifically between the collar and upper-arm, as well as between the torso and collar. These offsets preserve the coordinate system defined in Section II-A for computing pan-tilt inverse kinematics and allow the algorithms to be used in a modular fashion across multiple pan-tilt joints of the robot.

However, with the offsets, the coordinate frame *after* rotations through the shoulder pan-tilt module will no longer coincide with the coordinate frame *before* rotations through the elbow pan-tilt module, and similarly when considering *after* rotations through the waist pan-tilt and *before* rotating through the shoulder pan-tilt. The transformation matrix P_j accounts for the orientation offset and bridges the transition from one pan-tilt module to another.

Apart from the collar-upperarm and torso-collar pairs, the rest of the links do not have any orientation offsets, so transformation P_j is not required, i.e. $P_j = I$.

The pan-tilt and spherical joint modules require local task space information in order to compute the joint angles. Let the task space information ${}^0E_j$ be

$${}^0E_j = \begin{cases} {}^0X_{pos,j+1} - {}^0X_{pos,j} & \text{if pan-tilt module} \\ {}^0X_{orient,j} & \text{if spherical module} \end{cases} \tag{23}$$

where $j \in \{1, 2H, 2L, 3L, 4L, 2R, 3R, 4R\}$ is the index of the joint, $j + 1$ the index of the child joint, $X_{pos,j}$ is the position of joint j, and $X_{orient,j}$ the orientation of link j. Since ${}^0E_j$ is specified with respect to the global frame 0, it needs to be transformed to the local frame of the parent link $j-1$, as follows:

$$^{j-1}E_j \quad = \quad ^{j-1}R_0 \; ^0E_j \tag{24}$$

where $^jR_0$ is the orientation of the global frame 0 relative to the local frame of the jth link. This matrix can be computed incrementally by $^jR_0 = \; ^jR_{j-1} \; ^{j-1}R_{j-2} \; ...^1R_0$, where

$$^jR_{j-1} \quad = \quad (R_{m_j}P_j)^T \tag{25}$$

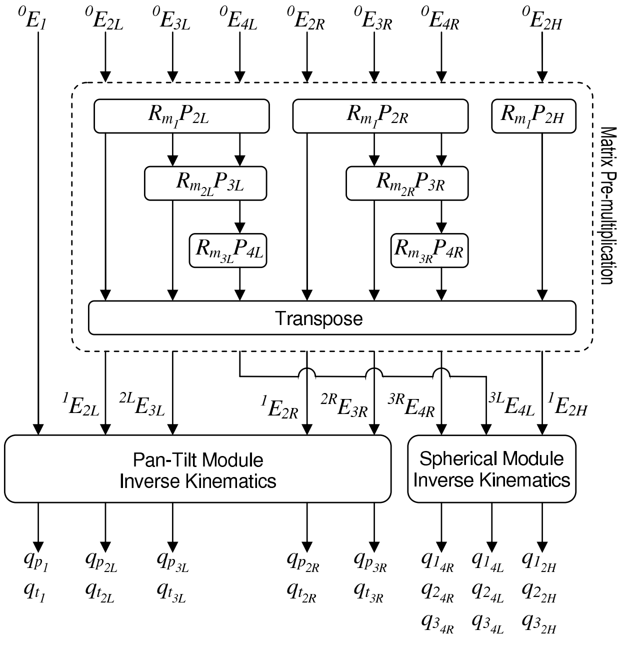
for $j \in \{1, 2H, 2L, 3L, 4L, 2R, 3R, 4R\}$, with

$$R_{m_j} = \begin{cases} R_p(q_{p_j}, q_{t_j}) & \text{if } j \in \{1, 2L, 3L, 2R, 3R\} \\ R_s(q_{1_j}, q_{2_j}, q_{3_j}) & \text{if } j \in \{2H, 4L, 4R\} \end{cases} \tag{26}$$

For example, to compute the joint angles for the left shoulder pan-tilt module, we first compute the position of the elbow relative to the shoulder in the coordinate frame of the collar, by transforming the global task frame to a local one, i.e.

$$\begin{aligned} ^1E_{2L} &= \; ^1R_0 \; ^0E_{2L} \\ &= (R_{m_0}P_1R_{m_1}P_{2L})^T \; ^0E_{2L} \\ &= (\text{rot}_\text{z}(q_{p_1})\text{rot}_\text{y}(q_{t_1})\text{rot}_\text{x}(90^\circ))^T \; ^0E_{2L} \end{aligned} \tag{27}$$

Then, we feed $^1E_{2L} = (x, y, z)^T$ into the pan-tilt inverse kinematics (12)-(13) and integrate $\dot{q}_p$ and $\dot{q}_t$ to obtain q_p and q_t respectively. Figure 4 shows a summary of the coordinate transformations required for each joint module.



$^0E_1$
$^0E_{2L}$
$^0E_{3L}$
$^0E_{4L}$
$^0E_{2R}$
$^0E_{3R}$
$^0E_{4R}$
$^0E_{2H}$
$R_{m_1}P_{2L}$
$R_{m_1}P_{2R}$
$R_{m_1}P_{2H}$
$R_{m_{2L}}P_{3L}$
$R_{m_{2R}}P_{3R}$
$R_{m_{3L}}P_{4L}$
$R_{m_{3R}}P_{4R}$
Matrix Pre-multiplication
Transpose
$^1E_{2L}$
$^{2L}E_{3L}$
$^1E_{2R}$
$^{2R}E_{3R}$
$^{3R}E_{4R}$
$^{3L}E_{4L}$
$^1E_{2H}$
Pan-Tilt Module Inverse Kinematics
Spherical Module Inverse Kinematics
q_{p_1} q_{t_1}
$q_{p_{2L}}$ $q_{t_{2L}}$
$q_{p_{3L}}$ $q_{t_{3L}}$
$q_{p_{2R}}$ $q_{t_{2R}}$
$q_{p_{3R}}$ $q_{t_{3R}}$
$q_{1_{4R}}$ $q_{2_{4R}}$ $q_{3_{4R}}$
$q_{1_{4L}}$ $q_{2_{4L}}$ $q_{3_{4L}}$
$q_{1_{2H}}$ $q_{2_{2H}}$ $q_{3_{2H}}$

Fig. 4. Module-to-module transformations for the entire robot.

III. Comparison with Jacobian Based Inverse Kinematics

In this section, we provide a simulation example of the proposed method and a comparison with closed loop Jacobian Based Inverse Kinematics (see e.g. [13]), described by

$$\dot{q} = J^*(\dot{X} - k_J e_X) \tag{28}$$

where k_J is a positive constant, J^* the right pseudo-inverse of the Jacobian $J = \partial X/\partial q$ regularized by a positive definite constant matrix Λ:

$$J^* = J^T(JJ^T + \Lambda) \tag{29}$$

and $e_X = \hat{X} - X$ the tracking error with $\hat{X} = (\hat{X}^T_{elbow}, \hat{X}^T_{wrist}, \hat{X}^T_{hand})^T$ computed via forward kinematics using recently estimated values of $q_1, ..., q_7$:

$$\begin{aligned} \hat{X}_{elbow} &= \text{rot}_\text{z}(q_1)\text{rot}_\text{y}(q_2)l_1e_1 \\ \hat{X}_{wrist} &= \text{rot}_\text{z}(q_1)\text{rot}_\text{y}(q_2) \\ &\quad \times (l_1e_x + P \; \text{rot}_\text{z}(q_3)\text{rot}_\text{y}(q_4)l_2e_3) \\ \hat{R} &= \text{rot}_\text{z}(q_1)\text{rot}_\text{y}(q_2)P \; \text{rot}_\text{z}(q_3)\text{rot}_\text{y}(q_4) \\ &\quad \times \text{rot}_\text{z}(q_5)\text{rot}_\text{y}(q_6)\text{rot}_\text{x}(q_7) \\ \hat{X}_{hand} &= (\hat{R}_{11}, \hat{R}_{21}, \hat{R}_{31}, \hat{R}_{12}, \hat{R}_{22}, \hat{R}_{32}, \hat{R}_{13}, \hat{R}_{23}, \hat{R}_{33})^T \end{aligned}$$

where $P = \text{rot}_\text{y}(90^\circ)$, $e_1 := (1, 0, 0)^T$, $e_3 := (0, 0, 1)^T$, and l_1, l_2 are the lengths of the upper and lower arms respectively.

In this study, we set $\Lambda = \text{diag}\{\lambda_i\}$, where $\lambda_i = 0.1$ for $i = 1, ..., 6$ and $\lambda_i = 10$ for $i = 7, ..., 15$, which provides fairly good tracking performance and robustness for comparison purpose.

For ease of illustration, we only consider a 7-DOF arm, with the shoulder as the origin of the global coordinate system. To evaluate joint tracking performance, we set sinusoidal desired joint trajectories to pass through 3 singularities: $q_{d_2} = -90^\circ$, $q_{d_4} = 0^\circ$, and $q_{d_6} = 90^\circ$. They are given by (in radians):

$$\begin{aligned} q_{d_1} &= 0.5\sin(0.01t) - \pi/4 \\ q_{d_2} &= 0.5\sin(0.01t) - \pi/2 \\ q_{d_3} &= 0.5\sin(0.02t) - 0.4 \\ q_{d_4} &= \pi/2\sin(0.01t) - \pi/4 \\ q_{d_5} &= 0.5\sin(0.01t) + \pi/4 \\ q_{d_6} &= 0.3\sin(0.015t) + \pi/2 - 0.3 \\ q_{d_7} &= 0.2\sin(0.01t) \end{aligned} \tag{30}$$

Then, we compute, via forward kinematics similar to (30), the desired task trajectories, which are then fed to the inverse kinematics algorithms to estimate the joint trajectories.

Figure 5 shows the joint angles estimated from the inverse kinematics algorithms, along with the actual joint angles. For the proposed Modular Inverse Kinematics, the estimated (solid) and actual (dotted) joint trajectories are closely matched everywhere except when near singularities, where a modest deviation is observed. However, for the Jacobian Based Inverse Kinematics, the deviations, some of which are substantial, are persistent and do not vanish even when away from singularities. Figure 6 shows that the joint tracking errors based on the proposed method are intermittent and less than 13° while those of the Jacobian Based Inverse Kinematics are persistent and reach as high as 37°.

That the tracking errors vanish when away from singularities is due to the adaptive regularization terms ε_{p_1}, ε_{p_2}, and ε_{s_1}, which only become active when near the singularities $q_2 = -90^\circ$, $q_4 = 0^\circ$, and $q_6 = 90^\circ$ respectively, as shown in

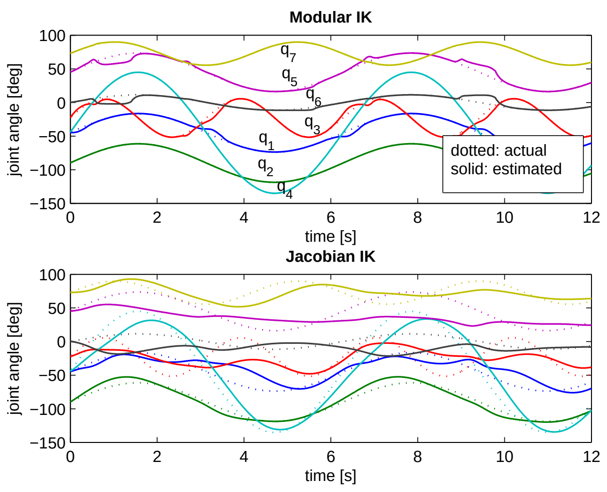


Fig. 5. Computed (solid) and actual joint angles (dotted) for proposed Modular Inverse Kinematics (IK) in comparison with Jacobian Based Inverse Kinematics.

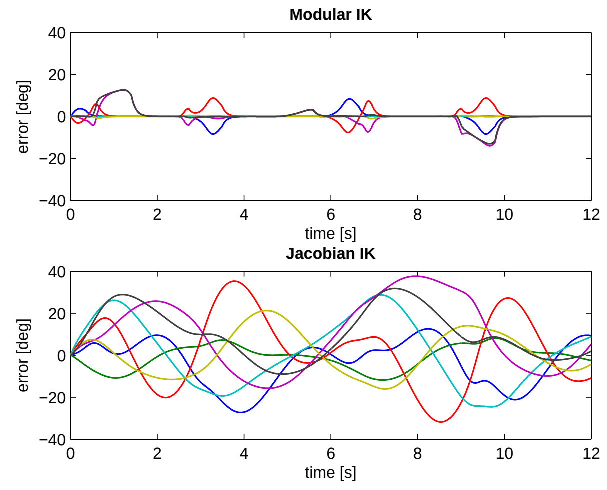


Fig. 6. Error between computed and actual joint angles.

Figure 8. Under the proposed framework, the regularization terms are easily and intuitively designed, based on task space information, according to (10). Although it is also possible to design adaptive regularization terms for the Jacobian based method based on [16], [17], [18], they require singular value decomposition of the Jacobian at every time step, which can be computationally intensive.

To investigate the robustness of the inverse kinematics algorithms to noise and singularities, we specified the following desired joint trajectories:

$$\begin{aligned} q_{d_i} &= 0.02(2p_i - 1), \quad i = 1,3,4,5,7 \\ q_{d_2} &= 0.02(2p_2 - 1) + \pi/2 \\ q_{d_6} &= 0.02(2p_6 - 1) - \pi/2 \end{aligned} \tag{31}$$

where $p_i \in [0,1]$, $i = 1,...,7$, are uniformly distributed random numbers. Thus, the robot is to track a noisy trajectory within a small neighborhood of 3 concurrent singularities. Figure 8 shows that the proposed Modular Inverse Kinematics is robust to noise at the singularities, and the tracking

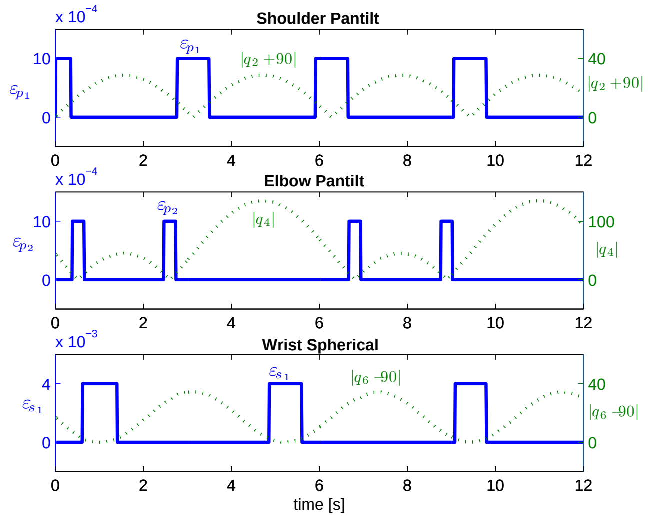


Fig. 7. Adaptive regularization terms become active only when near singularities.

error remains below $7°$. Although a gentle drift is observed over a long duration of $50s$, the same phenomenon is exhibited for Jacobian Based Inverse Kinematics. Fortunately, for turned based gesture imitation that we are dealing with, the duration of each trajectory is usually modest $(< 10s)$. Hence, the drift is unlikely to be a technical concern in our application.

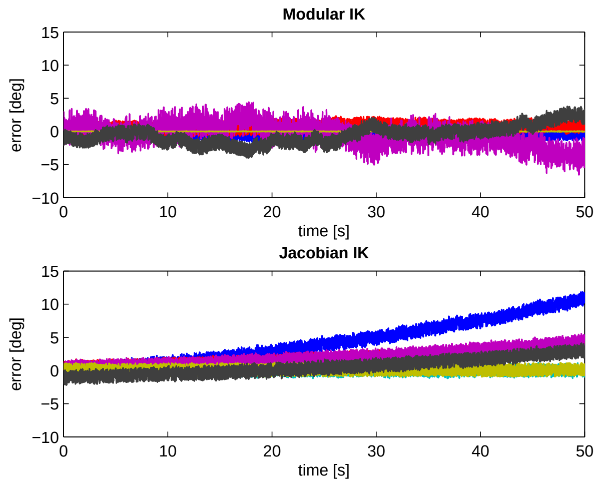


Fig. 8. The proposed Modular Inverse Kinematics (top) is robust to noise at singularity, although it exhibits a gentle drift, similar to Jacobian Based Inverse Kinematics (bottom).

A limitation of the proposed method is that it is not equipped to handle redundancy that may be present in general applications. For these purposes, Jacobian Based Inverse Kinematics provide a convenient and general solution. Nevertheless, for the specific application of gesture imitation, the proposed method is more advantageous, as shown in this comparison study.

IV. Handling Kinematic Constraints

We consider kinematic constraints in the form of joint limits $\underline{q}_i, \overline{q}_i$ and velocity limits $\pm\bar{\dot{q}}_i$, $i = 1, ..., n_{dof}$, which need to be satisfied simultaneously by the gesture trajectory. To this end, we define the admissible zones for joint position and velocity as:

$$\begin{aligned} \Omega_{q_i} &= \{q_i \in \mathbb{R} \ : \ \underline{q}_i \leq q_i \leq \overline{q}_i\} \\ \Omega_{\dot{q}_i} &= \{\dot{q}_i \in \mathbb{R} \ : \ |\dot{q}_i| \leq \bar{\dot{q}}_i\} \end{aligned} \tag{32}$$

such that $q_i, \dot{q}_i$ are to satisfy $q_i \in \Omega_{q_i}$ and $\dot{q}_i \in \Omega_{\dot{q}_i}$ for all $t \in [0, T]$, where T is the duration of the gesture.

Additionally, the modified trajectory is required to maintain its original profile as much as possible. Thanks to the turn-based imitation framework, batch modification of trajectory is feasible, and allows us to perform trajectory scaling while preserving the profile.

A. Position Scaling

Our method is based on *non-uniform* scaling of the joint position profile, such that relevant segments that transgress the joint limits are diminished while segments that remain within the limits are preserved. Additionally, smoothness is maintained despite the use of different scalings in different portions of the profile. Each joint is scaled independently.

First, we obtain the extrema of the segments of $q_i(t)$ that are outside the admissible zone, i.e. $q_i(t) \notin \Omega_{q_i}$ for some $t \in [0, T]$. For the kth extremum $(k = 1, ..., n_{e_i})$, we have:

$$q^*_{i,k} = \begin{cases} \max\limits_{t \in [\underline{c}_{i,k}, \overline{c}_{i,k}]} q_i(t), & \text{if } q_i(t) > \overline{q}_i \\ \min\limits_{t \in [\underline{c}_{i,k}, \overline{c}_{i,k}]} q_i(t), & \text{if } q_i(t) < \underline{q}_i \end{cases} \tag{33}$$

$$t^*_{i,k} = \begin{cases} \arg\max\limits_{t \in [\underline{c}_{i,k}, \overline{c}_{i,k}]} q_i(t), & \text{if } q_i(t) > \overline{q}_i \\ \arg\min\limits_{t \in [\underline{c}_{i,k}, \overline{c}_{i,k}]} q_i(t), & \text{if } q_i(t) < \underline{q}_i \end{cases} \tag{34}$$

for $t \in [\underline{c}_{i,k}, \overline{c}_{i,k}]$, where $\underline{c}_{i,k}, \overline{c}_{i,k}$ are the times at which the joint position leaves and re-enters, respectively, the admissible zone Ω_{q_i}.

Subsequently, we extract a segment occurring before $t^*_{i,k}$, such that it joins $q^*_{i,k}$ to the nearest turning point inside the admissible zone, or the nearest extremum outside the admissible zone, whichever is closer. Let $[\underline{t}^-_{i,k}, \overline{t}^-_{i,k}]$ be the time interval for this 'before-segment'. Define the sets:

$$\begin{aligned} \mathcal{T}_1 = \{t \in [t^*_{i,k-1}, t^*_{i,k}] \ : \ |\dot{q}_i(t)| < \epsilon, \ q_i(t) \leq \overline{q}_i\} \\ \mathcal{T}_2 = \{t \in [t^*_{i,k-1}, t^*_{i,k}] \ : \ |\dot{q}_i(t)| < \epsilon, \ q_i(t) \geq \underline{q}_i\} \end{aligned} \tag{35}$$

where $0 < \epsilon \ll 1$ is a small positive tolerance. Then, we have

$$\begin{aligned} \underline{t}^-_{i,k} &= \begin{cases} \max \ \Omega_{\epsilon_1}, & \text{if } \mathcal{T}_1 \neq \emptyset \text{ and } q^*_{i,k} > \overline{q}_i \\ \max \ \Omega_{\epsilon_2}, & \text{if } \mathcal{T}_2 \neq \emptyset \text{ and } q^*_{i,k} < \underline{q}_i \\ 0, & \text{if } \mathcal{T}_1 = \emptyset \text{ and } \mathcal{T}_2 = \emptyset \end{cases} \\ \overline{t}^-_{i,k} &= t^*_{i,k} \end{aligned} \tag{36}$$

where $t^*_{i,0} = 0$.

Similarly, we can extract a segment occurring after $t^*_{i,k}$. Let $[\underline{t}^+_{i,k}, \overline{t}^+_{i,k}]$ be the time interval for this 'after-segment'. Define the sets:

$$\begin{aligned} \mathcal{T}_3 = \{t \in [t^*_{i,k}, t^*_{i,k+1}] \ : \ |\dot{q}_i(t)| < \epsilon, \ q_i(t) \leq \overline{q}_i\} \\ \mathcal{T}_4 = \{t \in [t^*_{i,k}, t^*_{i,k+1}] \ : \ |\dot{q}_i(t)| < \epsilon, \ q_i(t) \geq \underline{q}_i\} \end{aligned} \tag{37}$$

Hence, we have

$$\begin{aligned} \underline{t}^+_{i,k} &= t^*_{i,k} \\ \overline{t}^+_{i,k} &= \begin{cases} \min \ \Omega_{\epsilon_3}, & \text{if } \mathcal{T}_3 \neq \emptyset \text{ and } q^*_{i,k} > \overline{q}_i \\ \min \ \Omega_{\epsilon_4}, & \text{if } \mathcal{T}_4 \neq \emptyset \text{ and } q^*_{i,k} < \underline{q}_i \\ 0, & \text{if } \mathcal{T}_3 = \emptyset \text{ and } \mathcal{T}_4 = \emptyset \end{cases} \end{aligned} \tag{38}$$

where $t^*_{i,n_{e_i}+1} = T$.

As such, each extreme point $q^*_{i,k}$ is associated with two segments – one before and one after. For convenience of representation, we define new labels for the segments as follows:

$$\begin{aligned} (t_{a_{i,j}} \ , \ t_{b_{i,j}}) &:= (\underline{t}^-_{i,k} \ , \ \overline{t}^-_{i,k}) \\ (t_{a_{i,j+1}} \ , \ t_{b_{i,j+1}}) &:= (\underline{t}^+_{i,k} \ , \ \overline{t}^+_{i,k}) \\ (q_{a_{i,j}} \ , \ q_{b_{i,j}}) &:= (q_i(t_{a_{i,j}}) \ , \ q_i(t_{b_{i,j}})) \end{aligned} \tag{39}$$

where $j = 1, ..., 2n_{e_i}$ and $k = 1, ..., n_{e_i}$.

Define a saturation function as:

$$S(\star) := \begin{cases} \overline{\star}, & \text{if } \star > \overline{\star} \\ \star, & \text{if } \underline{\star} \leq \star \leq \overline{\star} \\ \underline{\star}, & \text{if } \star < \underline{\star} \end{cases} \tag{40}$$

where $\overline{\star}$ and $\underline{\star}$ denote the upper and lower limits of $\star$, respectively. Then, the modified position q_{m_i} for the ith joint is obtained as follows:

$$q_{m_i}(t) = \begin{cases} s_{q_{i,j}}(q_i(t) - q_{a_{i,j}}) + S(q_{a_{i,j}}), & t \in [t_{a_{i,j}}, t_{b_{i,j}}] \\ & j = 1, ..., 2n_{e_i} \\ q_i(t), & \text{otherwise} \end{cases} \tag{41}$$

where

$$s_{q_{i,j}} = \frac{|S(q_{b_{i,j}}) - S(q_{a_{i,j}})|}{|q_{b_{i,j}} - q_{a_{i,j}}|} \tag{42}$$

is the scaling factor for the jth segment of the trajectory of joint i.

Since the interface between any two segments is always a turning point, where the derivative approaches zero, multiplying different scale factors for the two segments still results in the derivative approaching zero from both sides. This ensures smooth blending of the segments despite non-uniform scalings along the profile.

B. Velocity Scaling

A by-product of scaling down the joint positions to their limits is that the velocities $\dot{q}_{m_i}$ are automatically reduced too, so it makes sense to perform velocity scaling after joint scaling. Velocity scaling is performed in a similar way as joint scaling, with the exception that we apply *uniform* scaling across all joints over the entire movement. Let $\underline{d}_{i,k}, \overline{d}_{i,k}$ be the times at which the joint velocity leaves and re-enters, respectively, the admissible zone $\Omega_{\dot{q}_i}$. In addition, denote the set of times for which $\dot{q}_i(t) \notin \Omega_{\dot{q}_i}$ as follows:

$$\mathcal{T}_v := \{ \ t \in [\underline{d}_{i,k}, \overline{d}_{i,k}], k = 1, ..., n_{vex} \} \tag{43}$$

The scaling factor $0 < s_v \leq 1$ is a constant described by

$$s_v \quad = \quad \max_{i,j} \frac{\bar{\dot{q}}_i}{|\dot{q}^*_{i,k}|} \tag{44}$$

where $k = 1, ..., n_{ev_i}$, and

$$\dot{q}^*_{i,k} = \begin{cases} \max\limits_{t \in [\underline{d}_{i,k}, \overline{d}_{i,k}]} |q_i(t)|, & \text{if } t \in \mathcal{T}_v \\ \bar{\dot{q}}_i, & \text{otherwise} \end{cases} \tag{45}$$

The velocity scaling law (44) is simpler than its position scaling counterpart due to the symmetrical nature of the velocity constraints and the fact that each movement is discrete, i.e. zero initial and final velocities. Finally the modified trajectory is given by

$$q_{f_i}(t) = \int_0^{t/s_v} s_v \dot{q}_{m,i}(t) \ dt \tag{46}$$

which has an extended duration $(T/s_v) \geq T$ so that displacement is preserved.

V. Robotic Gesture Imitation Experiment

We have implemented the proposed inverse kinematics algorithms for gesture imitation on an upper body humanoid robot named *Olivia*, which has 17 DOFs in total: 6 for each arm, 2 for the torso, and 3 for the head. Olivia is a social robot designed mainly for human-robot communication and interactions using speech, vision, and gestures. To teach gestures to the robot, we use turn-based imitation, where the human completes a full demonstration before the robot performs a similar motion. This process emulates human-to-human imitation learning for intuitive teaching of gestures to robot without motion feedback from the robot, which may disturb human demonstration. Turn-based gesture imitation allows us to perform offline data processing, inverse kinematics, and motion replanning to satisfy kinematic constraints, before the final motion plan is executed by the motor controllers.

To capture human motion data in task space, we use the Measurand ShapehandPlus system, which is an upper body motion capture system comprising arrays of fibre optic bend and twist sensors to measure arm and hand movements, as well as inertial sensors to measure torso and head movements. Motion data is captured at 80Hz and consists of global Cartesian positions of the neck, shoulders, elbows and wrists, as well as global orientation Euler angles for the hands and head, all with respect to a reference frame located at a point on the hip. A motion segment from the human user is detected by using a velocity threshold, such that data is considered to be motion if any task velocity exceeds a threshold for a certain duration.

From the task space human trajectories, inverse kinematics based on the proposed method is performed to obtain the corresponding desired trajectories for the robot joints, and modified to satisfy kinematic constraints. Finally the desired motion is executed by a Proportional-Derivative controller.

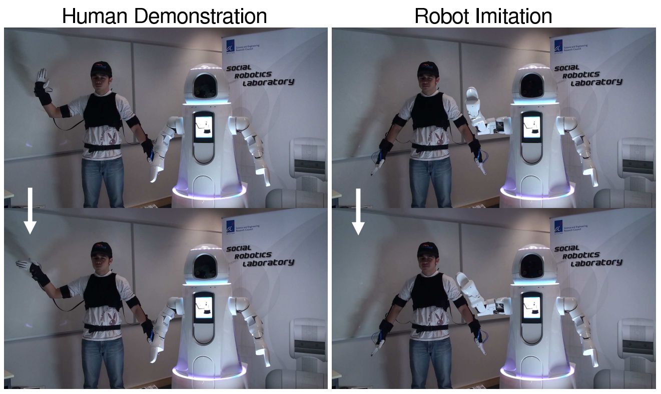


Fig. 9. Imitation of waving gesture.

The results of robotic imitation of waving, music-conducting, and forearm rotation gestures are shown in Figures 9-11 respectively. For each figure, the left column shows snapshots of the motion demonstrated by the user, whereas the right column shows the motion performed by the robot after the end of the demonstration. The imitated gestures are visually similar to the demonstrated ones, indicating that the proposed inverse kinematics scheme is effective. Figure 12 shows the the elbow flexion limit of $80°$ in the robot is respected when imitating a human gesture with elbow flexion $> 90°$. Turn-based imitation is useful when gestures involve the head, as seen in Figure 11, since it allows the demonstrator to focus on observing the robot's imitation motion after the demonstration is finished.

VI. Conclusions

We have presented a robust modular inverse kinematics scheme for an upper body humanoid robot performing gesture imitation, based on pan-tilt and spherical modules. We have designed adaptation laws for the joint angles based on the derivative of the inverse tangent function of the relevant task space variables. Robustness to kinematic singularity has been ensured via a time-varying regularization parameter, and kinematic constraints have been satisfied by using smooth scaling laws.

The proposed method is advantageous for applications such as gesture imitation where redundancy resolution is not required. The advantages include a better accuracy-robustness tradeoff with recovery of accuracy outside a neighborhood of the singularity, the ease of adapting the regularization parameter without relying on computationally expensive SVD operations, and avoidance of abrupt jumps in the derivative signal due to joint and velocity scaling.

Our simulation study has shown that the proposed method is robust to measurement noise and results in smaller overall error than a Jacobian based method with fixed damping. Furthermore, experimental implementation on an upper-body humanoid robot has shown that the robot replicates user-demonstrated gestures closely. The proposed robust modular

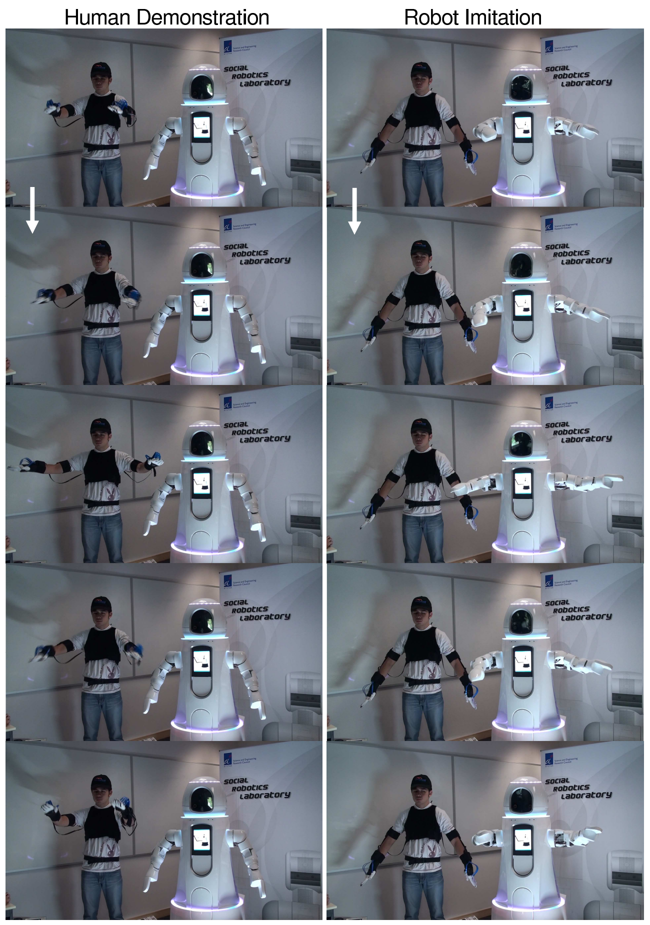


Fig. 10. Imitation of a music-conducting gesture.

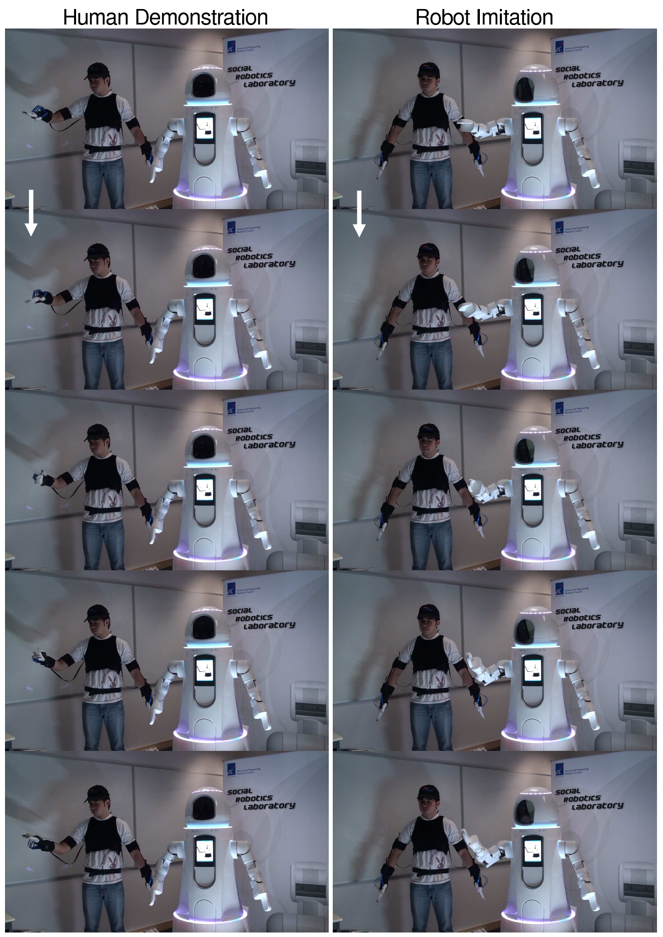


Fig. 11. Imitation of forearm rotation.

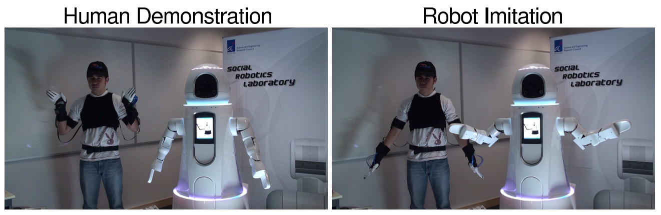


Fig. 12. Joint limit satisfaction.

inverse kinematics is useful for providing a natural and intuitive interface for a user to teach rich, highly coordinated, and possibly complex movements to a robot.

Acknowledgments

We thank Taiwen Chang, Jiayu Shen, Keck Soon Lim, and Hengxin Wu for their help in the practical implementation of the robotic system.

References

[1] N. Pollard, J. K. Hodgins, M. Riley, and C. Atkeson, “Adapting human motion for the control of a humanoid robot,” in *Proc. IEEE Intl. Conf. Robotics and Automation*, May 2002.

[2] S. Nakaoka, A. Nakazawa, and K. Yokoi, “Generating whole body motions for a biped humanoid robot from captured human dances,” in *Proc. IEEE Conf. Robotics & Automation*, (Taipei, Taiwan), pp. 3905–3910, 2003.

[3] S. Calinon and A. Billard, “Incremental learning of gestures by imitation in a humanoid robot,” in *Proc. ACM/IEEE Intl. Conf. Human-Robot Interactions*, pp. 255–262, 2007.

[4] R. Paul, B. Shimano, and G. Mayer, “Kinematic control equations for simple manipulators,” *IEEE Trans. Systems, Man, and Cybernetics*, vol. 11, no. 6, pp. 449–455, 1981.

[5] K. S. Fu, R. C. Gonzalez, and C. S. G. Lee, *Robotics Control, Sensing, Vision, and Intelligence*. McGraw Hill, 1987.

[6] J. Zhao and N. Badler, “Inverse kinematics positioning using nonlinear programming for highly articulated figures,” *ACM Transactions on Graphics*, vol. 13, pp. 313–336, 1994.

[7] D. Tolani, A. Goswami, and N. I. Badler, “Real-time inverse kinematics techniques for anthropomorphic limbs,” *Graphical Models*, vol. 62, pp. 353–388, 2000.

[8] M. Tarokh, K. Keerthi, and M. Lee, “Classification and characterization of inverse kinematics solutions for anthropomorphic manipulators,” *Robotics and Autonomous Systems*, vol. 58, pp. 115–120, 2010.

[9] D. E. Whitney, “Resolved motion rate control of manipulators and human prostheses,” *IEEE Transactions on Man-Machine Systems*, vol. 10, pp. 47–53, 1969.

[10] A. Balestrino, G. D. Maria, and L. Sciavicco, “Robust control of robotic manipulators,” in *Proc. 9th IFAC World Congress*, pp. 2435–2440, 1984.

[11] W. A. Wolovich and H. Elliot, “A computational technique for inverse kinematics,” in *Proc. 23rd IEEE Conference on Decision & Control*, pp. 1359–1363, 1984.

[12] Y. T. Tsai and D. E. Orin, “A strictly convergent real-time solution for inverse kinematics of robot manipulators,” *Journal of Robotics Systems*, vol. 4, pp. 447–501, 1987.

[13] P. Chiacchio, S. Chiaverini, L. Sciavicco, and B. Sciliano, “Closed-loop inverse kinematic schemes for constrained redundant manipulators with task space augmentation and task priority strategy,” *Intl. Journal of Robotics Research*, vol. 10, no. 4, pp. 410–425, 1991.

[14] Y. Nakamura and H. Hanafusa, “Inverse kinematic solution with singularity robustness for robot manipulator control,” *ASME Journal of Dynamical Systems, Measurement, & Control*, vol. 108, no. 3, pp. 163–

171, 1986.

[15] C. W. Wampler, “Manipulator inverse kinematic solutions based on vector formulations and damped least squares methods,” *IEEE Trans. Systems, Man, & Cybernetics*, vol. 16, no. 1, pp. 93–101, 1986.

[16] A. A. Maciejewski and C. A. Klein., “Obstacle avoidance for kinematically redundant manipulators in dynamically varying environments,” *Intl. Journal of Robotics Research*, vol. 4, p. 109117, 1985.

[17] S. Chiaverini, “Estimate of the two smallest singular values of the jacobian matrix: application to damped least-squares inverse kinematics,” *Journal of Robotics Systems*, vol. 10, no. 8, pp. 991–1008, 1993.

[18] S. Buss and J. S. Kim, “Selectively damped least squares for inverse kinematics,” *Journal of Graphics Tools*, vol. 10, no. 3, p. 3749, 2005.

[19] B. Dariush, M. Gienger, B. Jian, C. Goerick, and K. Fujimura, “Whole body humanoid control from human motion descriptors,” in *Proc. IEEE Conf. Robotics and Automation*, pp. 2677–2684, 2008.

[20] T. F. Chan and R. V. Dubey, “A weighted least-norm solution based scheme for avoiding joint limits for redundant joint manipulators,” *IEEE Trans. Robotics & Automation*, vol. 11, no. 2, pp. 286–292, 1995.