Real-time whole-body motion generation using torso posture regression and center of mass

Tsuichihara *et al. Robomech J (2018) 5:8*
https://doi.org/10.1186/s40648-018-0105-y

ROBOMECH Journal

RESEARCH ARTICLE

Open Access

Real-time whole-body motion generation using torso posture regression and center of mass

Satoki Tsuichihara*, Yuya Hakamata, Gustavo Alfonso Garcia Ricardez, Jun Takamatsu and Tsukasa Ogasawara

Abstract

For household humanoid robots, reaching as much workspace as possible with their hands is an important issue because the locations of target objects may range from the floor to above the robot's head. At the same time, to adapt to the constantly-changing household environment, inverse kinematics for the whole body must be solved in real time. In this paper, to achieve real-time motion generation for a humanoid robot, we propose a method of separating the inverse kinematics calculation into simpler problems. Using regression to estimate the torso orientation, we independently solve inverse kinematics for the lower body and both arms. First, using the target pose of both hands as input, we calculate the orientation of the torso and determine the target position of the center of mass considering the reachability of both arms. At each control step, we calculate the joint angles of the lower body from the position of the center of mass, feet poses, and torso orientation. Then, we calculate the joint angles of both arms. In experiments, we apply the proposed method to a human-size humanoid robot for reaching low-height positions while hunkering down. The proposed inverse kinematics solver is ten times faster than the numerical solution using the Jacobian matrix. We also verify the applicability of the proposed method using a sequence of random target positions for the hands as input.

Keywords: Coordinated movement, Whole-body motion, Humanoid robot, Visual feedback

Introduction

There are many research efforts aimed at enabling robots to perform household tasks such as cooking and serving food in daily-life environments [1–5]. Reaching target objects is the beginning of many such household tasks. Since the inputs of the reaching motion are the hands poses, it is necessary to calculate the whole body configuration of the robot (i.e., inverse kinematics (IK)). Cognetti et al. proposed whole-body IK and generate trajectories for a humanoid robot in a dynamics simulation environment [6]. They do not take care of the calculation time because they intend to use the proposed method in an offline manner.

For a robot to execute the sequences of reaching motions, these steps are repeated:

1. Determine the target configurations of the hands based on the current status of the environment,
2. Solve the whole body IK to satisfy these configurations,
3. Move the robot following the IK solution.

Unlike well-controlled environments in factories, household environments continuously change. Thus, the whole body IK should be solved every time. Since the robot stops moving until finishing the IK calculation, the calculation time can not be ignored.

To accelerate the calculation of IK, a number of issues must be considered, including:

i. The large number of DOF of a humanoid robot, and
ii. The robot's balance during execution.

Furthermore, the target positions to be reached can range from objects on the floor to those above the robot's head.

*Correspondence: satoki-t@rs.tus.ac.jp
Division of Information Science, Nara Institute of Science and Technology, 8916-5, Takayama, Ikoma 630-0192, Japan

As exemplified in Fig. 1, many household tasks require the robot to pick up objects at a very low height. Since it is very difficult to solve these issues in general, we concentrate on tasks where the robot keeps standing. Also, we ignore the collision avoidance issue (i.e., the robot moves around a large free space). We consider that the collision avoidance can be achieved by inputting appropriate trajectories of hands using a path planning technique.

We propose a method to accelerate the whole-body IK calculation when inputting the target configurations of both hands. To achieve a large reaching volume, we consider height changes, such as hunkering down. To solve the first issue (i) regarding the number of DoF, we split the whole-body IK problem into simpler IK subproblems. The idea is that if the torso posture is known, the remaining IK can be simplified by separately solving the IK for both arms and the lower body. To solve the second issue (ii) regarding the balance, we solve the IK considering the center of mass (CoM). Since the torso is almost fixed in our assumed tasks, we concentrate on tasks where the robot keeps standing and the motion is regarded as quasi-static. The meaning of the quasi-static is that COP position is similar to ZMP position.

In our method, first, we use regression to estimate the torso posture from the input position of the hands. Then, we solve the IK of the lower body to determine the torso posture and the target CoM. Finally, we solve the IK of each arm independently.

To prove the effectiveness of the proposed method, we perform experiments using a humanoid robot, the HRP-4 [7]. We verify that the proposed method calculates the IK ten times faster than the numerical solution, a Jacobian-based method, which is not satisfactory in calculating the IK for large DoF. In real-robot experiments, we verify that the proposed method generates whole-body motions to reach low-height positions while hunkering down and keeping balanced. Using our method, both the IK calculations and the robot's movements take three s on average. In the dynamics simulation experiments, we also verify the applicability of the proposed method to a sequence of several random target positions of the hands. In these simulations, the humanoid is standing and continuously reaches the targets while keeping its balance.

In this paper, we add two contributions that extend our previous work [8]. First, we design a method to calculate the target CoM position. We actually show the workspace volume of the proposed method. Second, we verify the applicability of the proposed method when target positions are sequentially input. In practice, it is unrealistic

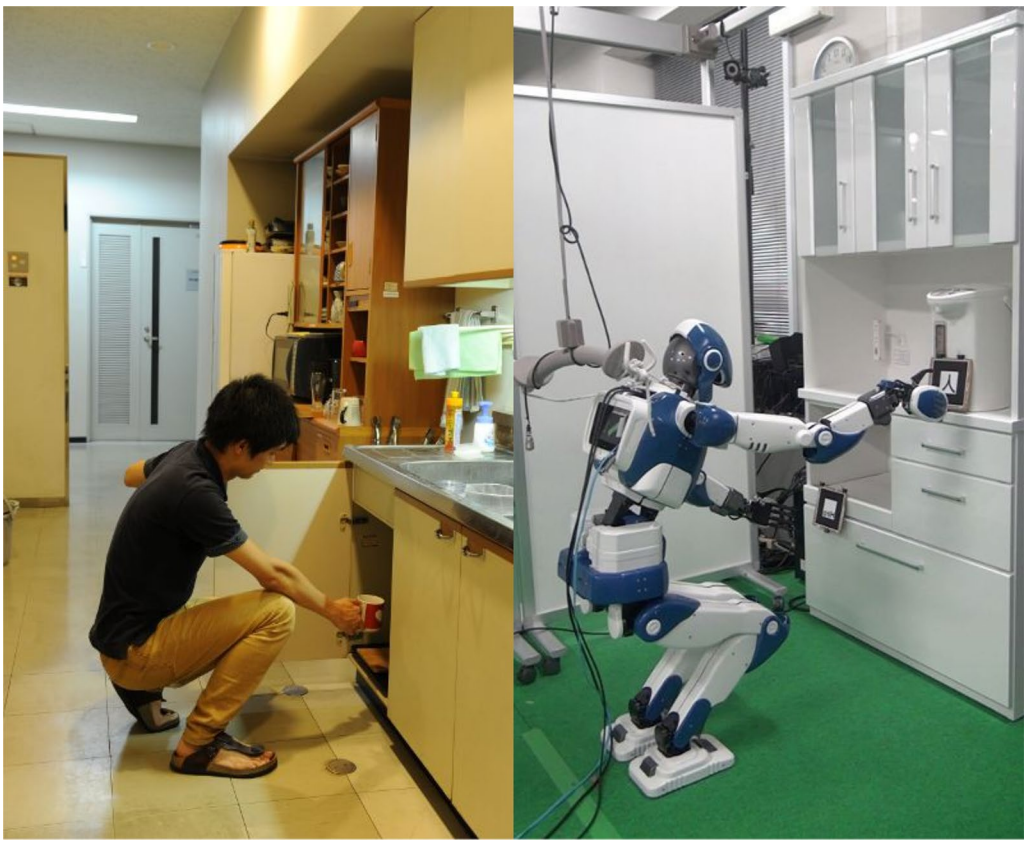

Fig. 1 Human (left) and humanoid robot (right) perform a manipulation task that requires hunkering down to reach the lower shelf in a kitchen

for a robot to move to the initial pose after reaching the target position every time. In the experiments, we apply the proposed method to a kitchen environment, where the robot needs to rotate the torso and squat.

The proposed method assumes that the humanoid robot meets the following conditions:

1. The robot changes the configuration of the waist by moving the legs.
2. The torso is connected to the waist by two chest joints (in the pitch and yaw directions).
3. Each arm has at least six DoF.

Most human-size humanoid robots satisfy these conditions. To consider the singularities, it is better to solve the IK of the arms analytically.

The rest of this paper is organized as follows. In "Related research" section, we introduce related research. In "Estimation of torso posture" section, we show how we calculate the torso posture using regression for real-time motion generation. In "Target CoM position generation" section, we describe an algorithm to define the target CoM position. In "Inverse Kinematics for the lower body" section, we explain how to solve the balance issue using the numerical method of IK. In "Implementation" Section, we show the implementation to achieve real-time whole-body motion. In "Results and discussion" section, we show the results and considerations of the whole-body motion generation. Finally, "Conclusions" section summarizes the contributions of this work.

Related research

Existing research efforts regarding motion generation [9, 10] can be roughly classified into two categories. The first category includes methods to refine a pre-defined motion to satisfy the target configuration, while the second category includes methods to generate the motion from scratch.

In the first category, Vannoy et al. proposed selecting one of multiple pre-defined motions considering the environment and a criterion of the motion quality [11]. Park et al. presented the use of non-linear optimization to adjust the pre-defined motions [12]. Otte et al. refined and repaired the pre-defined motions using a search-graph algorithm [13]. Since it is difficult for pre-defined motions to cover the whole range of target motions, the main purpose of these methods is real-time collision avoidance for sudden changes in the environment.

In the second category, Fok et al. used a numerical method (i.e., Jacobian-based) to solve the IK of the whole body [14]. Their main concern was implementing a middleware structure, not accelerating the calculation. Nishiwaki et al. proposed whole-body motion generation for reaching an object using one arm [15]. They tested their method with a real robot grasping an object on the floor. Nevertheless, they did not show the applicability of their method to reaching objects using both arms. Yamane et al. proposed a pin-and-drag interface for a human agent in a dynamics simulator [16]. They controlled the humanoid robot by pulling *links* in the dynamics simulation. Their method is suitable for computer graphics, where the animator controls the kinematics of the animated model. Ferrari *et al.* proposed a manipulation movement including locomotion for the humanoid robot [17]. They plan the CoM trajectory based on the distance to the target object and use the Jacobian-based whole-body IK to satisfy the planned CoM trajectory [18]. They tested their method in a dynamics simulation with a motion of reaching an object on a table. All these methods employ the standard Jacobian-based IK.

On the other hand, Khatib et al. proposed to use a potential field for real-time motion generation [19]. In this method, at every time step, the control input is decided from the derivatives of the field. Zucker et al. proposed to use distance fields for the trajectory formulation by optimizing a function that trades off between a smoothness component and an obstacle avoidance component [20]. Yang et al. proposed to use deep learning to determine the control input from an observed image [21]. They applied their proposed method to folding a towel. Unfortunately, they did not prove the applicability of their method for whole-body motion generation of humanoid robots. They applied their method to a 6-DoF manipulator in [19] and upper-body dual-arm robot in [21].

Our proposed method belongs to the second category. All methods in the second category use a Jacobian-based numerical method. Unlike all the previous methods, the proposed method reduces the calculation time by solving the center of the whole body (e.g., the torso) for managing the redundancy of the whole body well with a machine learning technique. Also, the methods in the first category are complementary to the second category. It is possible to use methods from the second category to modify the pre-defined motion. We expect that combining methods from the second category with the first category can enable the proposed method to deal with real-time collision avoidance.

Estimation of torso posture

For our proposed whole-body motion generation approach, we use regression to estimate the orientation of the torso. The inputs for this regression are the positions of both arms, i.e., (x_1, y_1, z_1) and (x_2, y_2, z_2). The outputs of this regression are the joint angles of the chest joint $(q_{\text{pitch}}, q_{\text{yaw}})$. If we assume that the orientation of the waist is the same as in the resting pose, we can obtain the torso orientation from the chest joint angles. We use the Support Vector Regression (SVR) [22] to calculate the orientation of the torso. The training dataset generation and the learning process are done offline, while in the actual robot, the motion generator uses this regression online.

We use Algorithm 1 to generate the dataset, which implements a numerical solution of the IK. We select the robot's configurations that satisfy the positions of both arms, and then use only the chest joints' angles for the dataset. During learning, the height of the torso is fixed and only the chest joints are learned. In this paper, we assume that the workspace is in front of the humanoid robot, and define the initial posture for the numerical IK, as shown in Fig. 2. As indicated by the yellow boxes in Fig. 2, the robot's workspace has 0.45 m in the x (frontal) axis, 0.8 m in the y (lateral) axis, and 0.75 m in the z (vertical) axis from the initial posture.

Algorithm 1 Dataset generation

Input: $\boldsymbol{q}_0, \boldsymbol{p}^*_{\text{left}}, \boldsymbol{R}^*_{\text{left}}, \boldsymbol{p}^*_{\text{right}}, \boldsymbol{R}^*_{\text{right}}$
Output: $\boldsymbol{p}^*_{\text{left}}, \boldsymbol{p}^*_{\text{right}}, \boldsymbol{q}_{\text{chest}}$

1: $\boldsymbol{q} \leftarrow \boldsymbol{q}_0$
2: **while** $isSolvable$ **do**
3: $\quad \boldsymbol{q}, isSolvable \leftarrow BothArmsIK\left(\boldsymbol{q}, \boldsymbol{p}^*_{\text{left}}, \boldsymbol{R}^*_{\text{left}}, \boldsymbol{p}^*_{\text{right}}, \boldsymbol{R}^*_{\text{right}}, \omega\right)$
4: $\quad \omega \leftarrow \omega + 1.0$
5: **end while**
6: $\boldsymbol{q}_{\text{chest}} \leftarrow \left(q_{\text{chest}_{\text{pitch}}}, q_{\text{chest}_{\text{yaw}}}\right) \in \boldsymbol{q}$

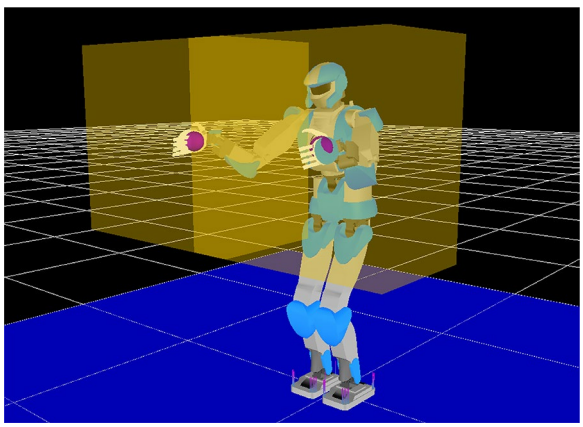

Fig. 2 The initial posture of the humanoid robot for the learning procedure, with yellow boxes indicating the humanoid's workspace and purple spheres indicating the position of the end effectors

Using the Levenberg–Marquardt (LM) method [23], the numerical method of the IK *BothArmsIK* (step 3 of Algorithm 1) with initial joint angles $\boldsymbol{q}_0$ solves the joint angles in the upper body for the target pose of both arms. In the LM method, ω represents the value of the diagonal matrix added to the diagonal elements of the Jacobian matrix. We improve the solvability of the IK near the singularity points of the end effector by increasing the ω value in an exploratory way. The boolean value *isSolvable* indicates whether this IK solution converges.

To test the learned regression, we apply several target positions. Figure 3 shows the postures of the humanoid obtain with the generated regression, and the purple spheres indicate the target positions of both hands as the inputs to the regression. Table 1 shows the three parameters we use for the regression algorithm.

Target CoM position generation

In a humanoid robot, changing the height of the body using their legs increases the manipulation area. To reduce the calculation cost, we select the target position of the CoM of the whole body from three candidates, considering the reachability of both arms. Figure 4 shows the three candidates and the overlapping workspaces of this approach.

We determine the reachability of humanoid robots using the position of both hands' targets ($\boldsymbol{p}^*$) and both shoulders ($\boldsymbol{p}_{\text{shoulder}}$) in the initial configuration of the robot. For reachability, the following condition should be satisfied:

$$\left|\boldsymbol{p}^* - \boldsymbol{p}_{\text{shoulder}}\right| < L_{\text{arm}}. \tag{1}$$

The length L_{arm} represents the length from the shoulder joint to the wrist of the robot. If neither arm can reach the target position in the current configuration, we set the target position of the CoM ($\boldsymbol{p}^*_{\text{CoM}}$) as

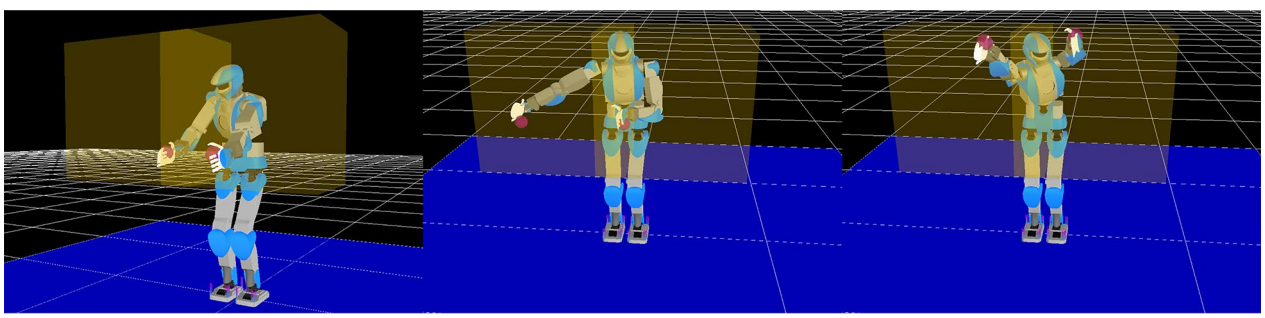

Fig. 3 Humanoid's postures obtained from the SVR and the inverse kinematics of the arms

Table 1 Parameters for the regression algorithm

Parameter	Value
Kernel width	0.1
Constraint	1.0
Slack variable	0.0001

$$\boldsymbol{p}^*_{\mathrm{CoM}} \leftarrow \begin{cases} \boldsymbol{p}_{\mathrm{CoM_{init}}} + \boldsymbol{b} & \left(p^*_z - p_{\mathrm{shoulder_z}} > L_{\mathrm{arm}}\right), \\ \boldsymbol{p}_{\mathrm{CoM_{init}}} - \boldsymbol{b} & \left(p^*_z - p_{\mathrm{shoulder_z}} < -L_{\mathrm{arm}}\right), \\ \boldsymbol{p}_{\mathrm{CoM_{init}}} & (\text{otherwise}). \end{cases} \quad (2)$$

Using this equation, a vector of the CoM position $\boldsymbol{p}_{\mathrm{CoM_{init}}}$ is calculated from the initial configuration of the robot, where the vector $\boldsymbol{b}$ is a pre-defined displacement $(0, 0, 0.1)$ in the coordinate system of the robot. If the target hand position is set to a location far from the shoulder, Eq. 2 determines the necessary change in the CoM position ($\boldsymbol{p}^*_{\mathrm{CoM}}$) to a higher or lower position. Note that the CoM trajectory is calculated by interpolating the initial and the target CoM positions.

Inverse Kinematics for the lower body

We use Algorithm 2 to calculate the joint angles of both legs. We control the CoM of the whole body, the torso's orientation, and the poses of the legs. The CoM of the whole body is calculated by assuming the configuration of the upper body (e.g., the configuration in the previous time step). We use the numerical IK to solve the joint angles of the lower body. Since the number of DoF of the lower body is a subset of the DoF of the whole body, this calculation is faster than the IK of the whole body.

Algorithm 2 Inverse Kinematics for both legs

Input: $\boldsymbol{q}_0, \boldsymbol{p}^*_{\mathrm{CoM}}, \boldsymbol{R}^*_{\mathrm{torso}}, \boldsymbol{p}^*_{\mathrm{swing}}, \boldsymbol{R}^*_{\mathrm{swing}}$
Output: $\boldsymbol{q}_{\mathrm{legs}}$

$\boldsymbol{q}_{\mathrm{legs}}, \boldsymbol{q}_{\mathrm{upperBody}} \leftarrow \boldsymbol{q}_0$
for $i = 0$ to N **do**
  $\boldsymbol{p}_{\mathrm{CoM}}, \boldsymbol{R}_{\mathrm{torso}}, \boldsymbol{p}_{\mathrm{swing}}, \boldsymbol{R}_{\mathrm{swing}}, \boldsymbol{J} \leftarrow \mathrm{FK}\left(\boldsymbol{q}_{\mathrm{legs}}, \boldsymbol{q}_{\mathrm{upperBody}}\right)$
  $\boldsymbol{e} \leftarrow \begin{pmatrix} \boldsymbol{p}^*_{\mathrm{CoM}} - \boldsymbol{p}_{\mathrm{CoM}} \\ \mathrm{LN}\left(\boldsymbol{R}^\top_{\mathrm{torso}} \boldsymbol{R}^*_{\mathrm{torso}}\right) \\ \boldsymbol{p}^*_{\mathrm{swing}} - \boldsymbol{p}_{\mathrm{swing}} \\ \mathrm{LN}\left(\boldsymbol{R}^\top_{\mathrm{swing}} \boldsymbol{R}^\star_{\mathrm{swing}}\right) \end{pmatrix}$
  if $|\boldsymbol{e}| < tol$ **then**
    break
  end if
  $\boldsymbol{q}'_{\mathrm{legs}} \leftarrow \boldsymbol{q}_{\mathrm{legs}} + \alpha \boldsymbol{J}^\sharp \boldsymbol{e}$
  $\boldsymbol{p}'_{\mathrm{CoM}}, \boldsymbol{R}'_{\mathrm{torso}}, \boldsymbol{p}'_{\mathrm{swing}}, \boldsymbol{R}'_{\mathrm{swing}}, \boldsymbol{J}' \leftarrow \mathrm{FK}\left(\boldsymbol{q}'_{\mathrm{legs}}, \boldsymbol{q}_{\mathrm{upperBody}}\right)$
  $\boldsymbol{e}' \leftarrow \begin{pmatrix} \mathrm{LN}\left(\boldsymbol{R}'^\top_{\mathrm{torso}} \boldsymbol{R}^*_{\mathrm{torso}}\right) \\ \boldsymbol{p}^*_{\mathrm{swing}} - \boldsymbol{p}'_{\mathrm{swing}} \\ \mathrm{LN}\left(\boldsymbol{R}'^\top_{\mathrm{swing}} \boldsymbol{R}^\star_{\mathrm{swing}}\right) \end{pmatrix}$
  $\boldsymbol{q}_{\mathrm{legs}} \leftarrow \boldsymbol{q}'_{\mathrm{legs}} + \beta \boldsymbol{J}'^\sharp \boldsymbol{e}'$
end for

In Algorithm 2, we assume that the robot's sole of the support leg is on the ground, and the coordinate system of the robot is based in sole of the support leg. If the robot stands on both legs, we simply choose one of the legs as the support leg, and define the other leg as the swing leg. In this algorithm, the vector $\boldsymbol{q}_{\mathrm{legs}} = \left(\boldsymbol{q}_{\mathrm{right}}, \boldsymbol{q}_{\mathrm{left}}\right)$ represents the lower-body joint angles, that is, the joint angles of the right and left legs. The vector $\boldsymbol{q}_{\mathrm{upperBody}}$ represents the upper-body joint angles, that is, the joint angles of both arms and the chest. The initial joint angles of the whole body are $\boldsymbol{q}_0$. The CoM position of the whole body is denoted by $\boldsymbol{p}_{\mathrm{CoM}}$ (see "Target CoM position

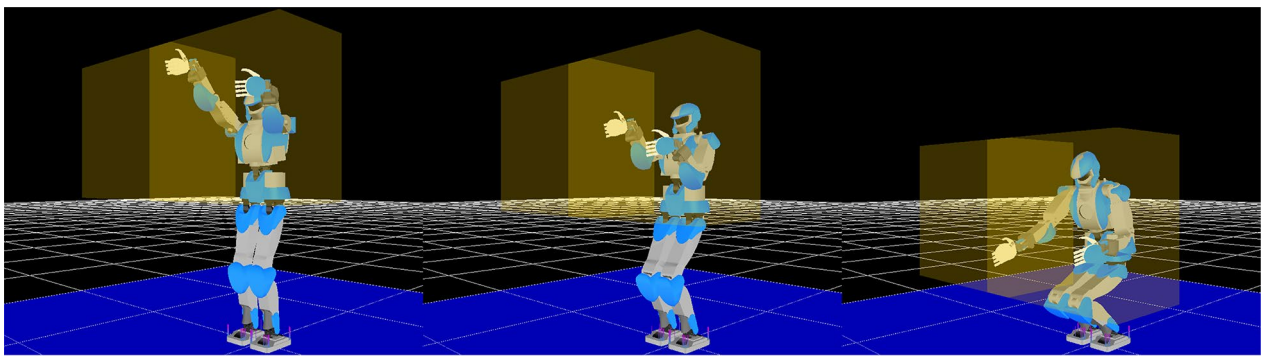

Fig. 4 Reachable areas (yellow boxes) of the proposed method in the three configurations of the CoM (Eq. 2)

generation" section), while $\boldsymbol{p}_{\text{swing}}$ and $\boldsymbol{R}_{\text{swing}}$ are the position and the orientation of the foot sole in the swing leg, $\boldsymbol{R}_{\text{torso}}$ is the orientation of the torso link, and $\boldsymbol{J}$ is the Jacobian matrix of the lower body. The vector $\boldsymbol{x}^*$ is the target value of $\boldsymbol{x}$, and $\boldsymbol{Y}^\sharp$ is the pseudo-inverse of the matrix $\boldsymbol{Y}$. FK is a forward kinematics function. The input of FK is the joint angles $\boldsymbol{q}$ and the outputs are the position $\boldsymbol{p}$ and orientation $\boldsymbol{R}$ of the CoM, torso and swing leg, and the Jacobian of the lower body $\boldsymbol{J}$. The function LN converts the rotation matrix $\boldsymbol{R}$ to an angular vector [24]. The value N is the number of maximum iterations. The value *tol* is the tolerance for the norm of the error vector $\boldsymbol{e}$. The error vector has 12 dimensions that consist of the CoM positions, the torso orientation, and the position and orientation of the swing leg. Using the Jacobians $\boldsymbol{J}_{\text{CoM}}$, $\boldsymbol{J}_{\text{torso}}$ and $\boldsymbol{J}_{\text{swing}}$ for the CoM, the torso and the sole of the swing leg, the Jacobian can be represented as

$$\boldsymbol{J} = \left[\boldsymbol{J}_{\text{CoM}}\ \boldsymbol{J}_{\text{torso}}\ \boldsymbol{J}_{\text{swing}}\right]^\top. \tag{3}$$

Implementation

The system consists of four major parts: the online localization, the posture estimator, the whole-body controller, and the robot controller, as shown in Fig. 5. In this figure, the arrows indicate the data flow. Object localization and recognition is out of the scope of this paper, so we use AR markers for simplification. We use the ARToolKit library [25] to estimate the position of the AR markers and their ID's.

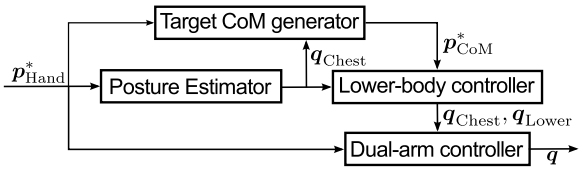

Fig. 6 The data flow in both the posture estimator and the whole-body controller (the target CoM generator, the posture estimator, the lower-body controller, and the dual-arm controller)

Figure 6 shows the data flow between the posture estimator and elements of the whole-body controller. First, the target CoM generator determines the target CoM position corresponding to both hands at the target. After updating the value of the chest joint from the posture estimator, the lower-body controller calculates the joint angles in the chest and the legs' joints. Finally, the dual-arm controller calculates the joint angles of both arms.

Generating motion using inverse kinematics

To generate the motion, we calculate the intermediate hand configurations by linear interpolating the initial value and the target value. To interpolate the orientation, we multiply the initial rotation matrix and the rotation matrix calculated from the Rodrigues equation using the vector of the angular velocity from the difference between the target value and the initial value.

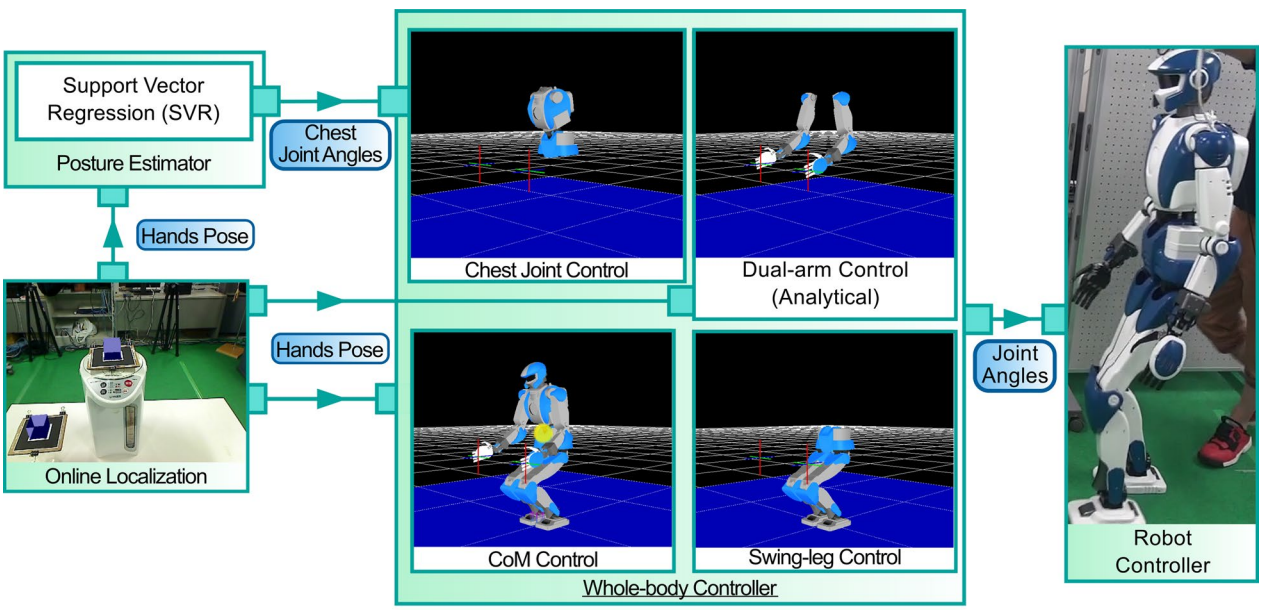

Fig. 5 System overview. Arrows indicate the data flow. The system is formed by the online localization, the posture estimator, the whole-body controller, and the robot controller

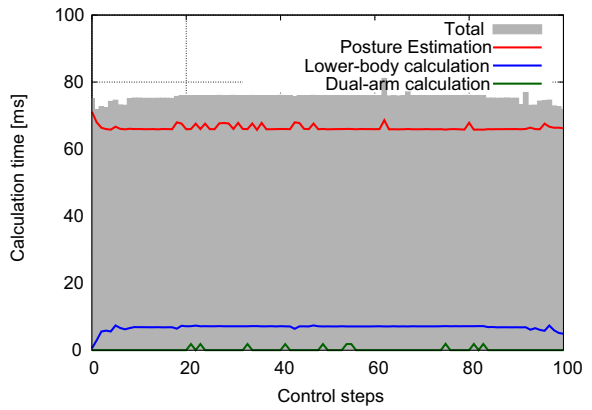

Fig. 7 The calculation time of SVR, lower-body IK and both arms IK. Most of the calculation time corresponds to the SVR calculation

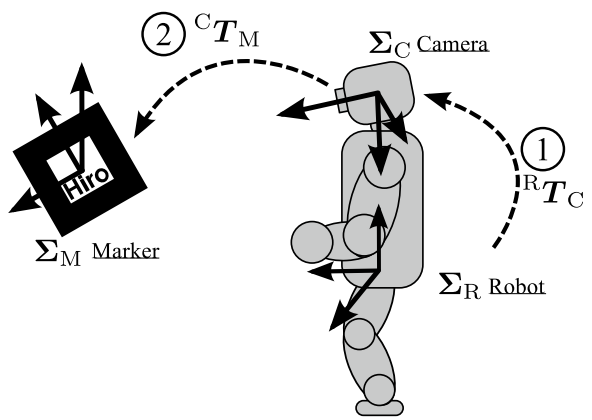

Fig. 8 The relationships between each coordinate system

Posture estimator

We select the Gaussian kernel as the kernel function of the SVR. In this research, we use the regression only once when inputting the target positions of the end effectors, and then we interpolate the results instead of calculating the regression at every control step. For the actual robot implementation, the robot's CPU (Intel Pentium M Processor 1.6 GHz) has a very low clock frequency, so it is slow to calculate the regression. Figure 7 shows the calculation times for the regression, the numerical IK for the lower body, the analytical IK for both arms, and the total calculation time. The mean values of the calculation times over 100 samples are: SVR, 66.4 ms; lower body, 6.8 ms; analytical inverse kinematics, 0.2 ms; and total, 75.5 ms. It turns out that the most time-consuming process is the SVR calculation.

Measuring the 3D position of each object

To manipulate an object located by the ARToolKit, we determine the hand pose following these steps:

1. Estimate the position of each object in the coordinate system of the camera mounted on the robot, and
2. Translate from the coordinate system of the camera to that of the robot.

Figure 8 shows the three coordinate systems of the robot $\boldsymbol{\Sigma}_{\mathrm{R}}$, the camera $\boldsymbol{\Sigma}_{\mathrm{C}}$, and the marker $\boldsymbol{\Sigma}_{\mathrm{M}}$. The matrices ${}^{\mathrm{R}}\boldsymbol{T}_{\mathrm{C}}$, ${}^{\mathrm{C}}\boldsymbol{T}_{\mathrm{M}}$, and ${}^{\mathrm{M}}\boldsymbol{T}_{\mathrm{O}}$ are homogeneous transformations from the coordinate system of the robot to the camera, from the camera to the marker, and from the marker to the object, respectively. The homogeneous transformation matrix from the coordinate system of the robot to the object is computed as

$$ {}^{\mathrm{R}}\boldsymbol{T}_{\mathrm{O}} = {}^{\mathrm{R}}\boldsymbol{T}_{\mathrm{C}}\,{}^{\mathrm{C}}\boldsymbol{T}_{\mathrm{M}}\,{}^{\mathrm{M}}\boldsymbol{T}_{\mathrm{O}}. \tag{4} $$

The matrix ${}^{\mathrm{R}}\boldsymbol{T}_{\mathrm{C}}$ is calculated using forward kinematics. We know ${}^{\mathrm{M}}\boldsymbol{T}_{\mathrm{O}}$ from the location of the marker on the object.

Results and discussion

Comparing the calculation times using a dynamics simulation

We show the effectiveness of the whole-body controller using the humanoid robot HRP-4 in the dynamics simulator OpenHRP [26]. As a conventional method, we use the LM method for the numerical solution of IK [23], as described in "Estimation of torso posture" section. In the conventional method, the Jacobian matrix has 28 rows corresponding to the DoF and 24 columns corresponding to the position of both hands, the CoM, and the orientation of both hands, the torso, and the swing leg. As shown in Fig. 9, the target position of both hands is in front of a table. In the numerical calculation of both methods, we configured the tolerance (*tol* in Algorithm 2) to 1×10^{-6} m. Also, we set the maximum number of iterations to 100 and the coefficients of the update function (α, β in Algorithm 2) to 0.3.

Figure 9 shows the postures of the HRP-4: the initial posture (a), the interpolated posture (b), and the posture after the movement (c). We apply the proposed and conventional methods for 200 interpolated points from the initial to the final configurations. In the setting of Fig. 9, the length between the initial configuration (a) and the final configuration (c) is 0.4 m, and the distance between each intermediate position is 1.9 mm. Assuming that the control cycle is 5 ms, the number of intermediate points is calculated for both the end effector and CoM with an acceptable speed of 0.38 m/s in the proposed and conventional methods.

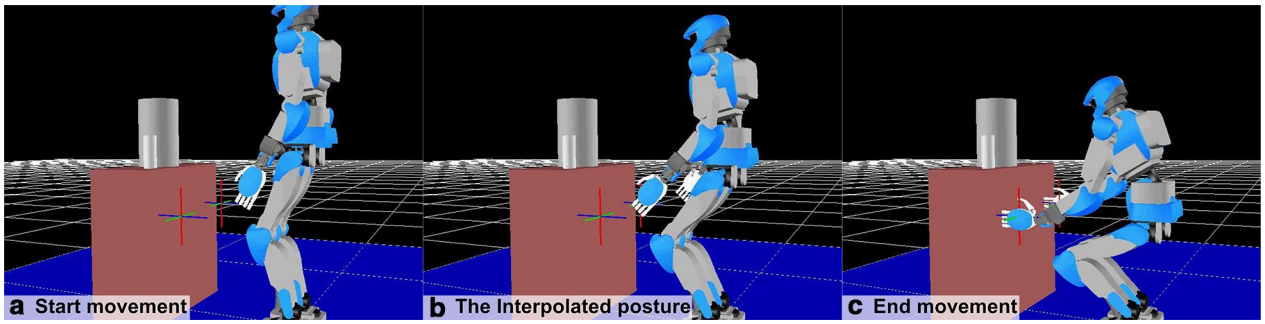

Fig. 9 The posture of HRP-4 generated by the proposed method. HRP-4 completely reaches the target poses (three-colored axes) with the end effectors (**a** start movement, **b** interpolated posture, and **c** end movement)

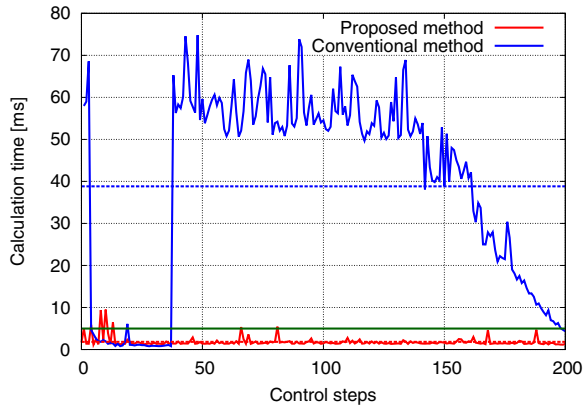

Fig. 10 Comparison of the calculation times. In the control steps 10 to 30, the conventional method calculates the joint angles rapidly because the robot's pose is similar to the initial pose

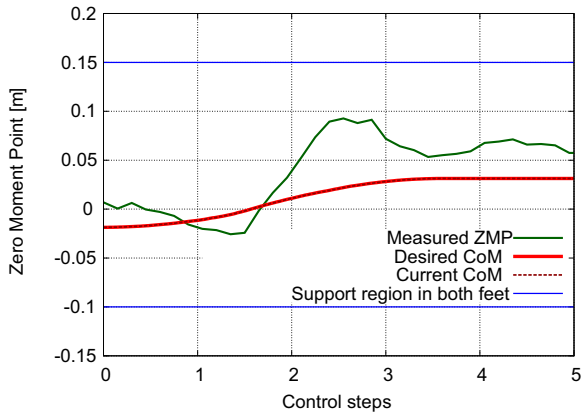

Fig. 11 The ZMP of both feet and CoM during the whole-body movement. The support region between the two blue lines represents the area from the toe to the heel

Figure 10 shows the comparison of the calculation times. Using the proposed method, the mean of the calculation times is 1.8 ms, and, using the conventional method, 38.8 ms. Since the target position of both hands is further from the singularities, the calculation time of the conventional method is smaller. We also verified that, in general, the calculation of the proposed method was faster than the conventional method. However, as shown in Fig. 10, from control step 10 to 30, the conventional method rapidly calculates the joint angles. As can be seen in Fig. 9, the postures (a) and (b) are very similar (near the initial pose), so the conventional controller takes a short time.

Figure 11 shows the Zero Moment Point (ZMP) using a moving average filter and the CoM position when moving to a hunkering-down posture using the proposed method. The upper blue line shows the position of the toe, and the lower blue line represents the heel. The area between the two lines represents the support region of the HRP-4's feet. The trajectory of the ZMP is always inside the support region during this movement. This result shows that the movement generated by our method is well-balanced. We also show the current and desired values of the CoM produced by the proposed method. The trajectories of the CoM are the same during this movement, so the proposed controller is accurate.

As shown in Fig. 12, we evaluate the accuracy of the proposed method with the error of the CoM position and the end effector. The error of the end effector position is less than 1×10^{-15} m.

Evaluation of the proposed method

We evaluate the applicability of the proposed method for whole-body movement using a numerical method as the conventional method for comparison. As shown in Fig. 3, we test both methods with 729 (27 poses for each hand) random target poses in the workspace (Fig. 2). In both methods, we also use linear interpolation to evaluate these poses. Table 2 shows the results using both methods for all target poses. The error norm is the average

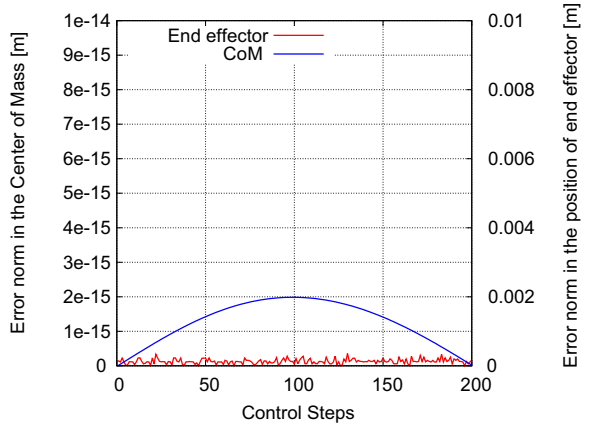

Fig. 12 The error norm of the CoM position and the end effector during the whole-body movement

Table 2 Comparison of results in both methods

	Error norm (mm)	Calculation time (ms)	Total time (s)
Proposed	0.62	1.1	1.0
Conventional	9.99	46.4	2.6

norm of the difference vector between the target pose and the current pose of both hands. The calculation time is the average time to calculate the whole-joint angles at each control step. The total time is the average time for moving the whole body in the dynamics simulator.

As can be seen in Table 2, the results of the error norm and the calculation time with the proposed method are better than with the conventional method. Using the conventional method, the error norm is large when the target positions are on the far side in the lateral direction. The conventional method can solve the whole-body joint angles including the chest joint, but it is difficult to reduce the error norm with the torso posture. The proposed method is faster in such cases because it uses the trained data for the calculation of the torso posture. Using the proposed method, the total time for movement is around 38% of the time using the conventional method.

Motion generation experiment using the real robot

We also verify the effectiveness of the proposed whole-body controller using an actual HRP-4. In this experiment, the target positions of both hands are obtained from the AR markers, which are observed by the camera mounted on the head of the HRP-4. Figure 13 shows the images captured from the HRP-4's camera and processed by the ARToolKit.

Figure 14 shows the generated movement of the HRP-4 from the initial posture (a), through the interpolated postures (b) and (c), and to the posture after the movement (d). The HRP-4 reaches the two AR markers with both hands, while maintaining its balance. This movement takes 3 s. A video with the humanoid reaching the AR markers is attached to this publication (Additional file 1).

Sequential whole-body control

We apply the proposed method to the HRP-4 in the dynamics simulator for sequentially approaching several positions. Figure 15 shows the poses of the humanoid robot approaching ramdomly generated target positions. The three color arrows in Fig. 15 indicate the target positions input to the HRP-4. In this figure, the HRP-4 successfully reaches three target positions using its whole body, while maintaining its balance. Figure 15a depicts the motion of the HRP-4 to reach target positions above its head. Figure 15b, c show cases where one of the hands is reaching a high position, while the other is reaching a low position. A video with the humanoid reaching these positions is attached to this publication (Additional file 2). These figures and videos demonstrate the applicability of the proposed method to whole-body control with sequentially random inputs.

In Fig. 4, we can see the movable area using the proposed method. Comparing both Figs. 4 and 15, a large workspace (0.45 m $\times$ 0.80 m $\times$ 1.25 m) and the applicability of the proposed method is verified.

Application to a living environment

We also apply the proposed method to a human living environment using a dynamics simulator. Figure 16 shows the pose of the humanoid robot approaching two target positions on a table and a shelf. In this experiment, we solve the collision avoidance issue by assigning waypoints to the hands' trajectories. Figure 16a shows the initial pose of the HRP-4. Figure 16c and f show the HRP-4 approaching the objects with both hands. In Fig. 16d, the HRP-4 is moving to the next target position on the upper shelf.

It takes 3 s to approach the position on the table and 5 s to approach the position on the upper shelf (Additional file 3). Because we set the waypoints for each target configuration, it takes longer to approach the position than using the motion planning without a waypoint. Multiple movements including chest rotation and squatting are successfully generated using the proposed method, and the time of the movement is reasonable for users.

Our proposed method uses only the CoM for keeping the robot's balance. Using our method, the robot can not reach an edge of the ZMP's support polygon with a fast whole-body movement. However, when the robot's balance is close to this edge, the robot should move slowly

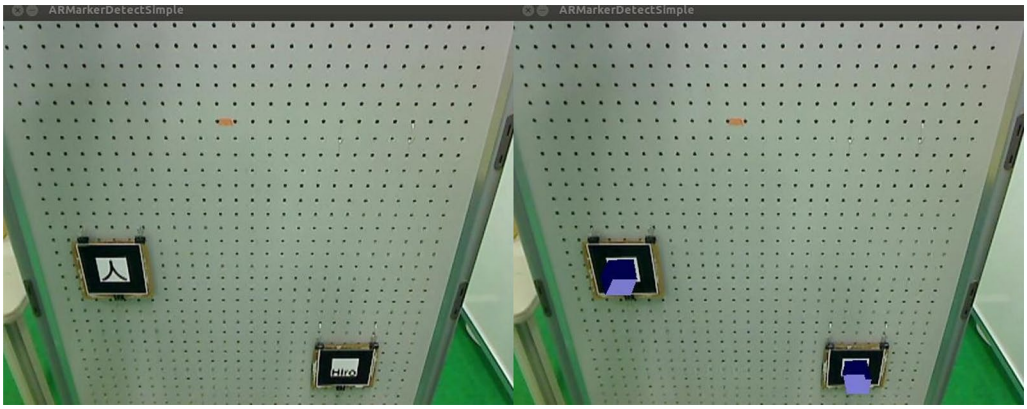

Fig. 13 Estimation of the marker position using the ARToolKit library (left: input image from the HRP-4's camera, right: rendered blue cubes at the estimated positions)

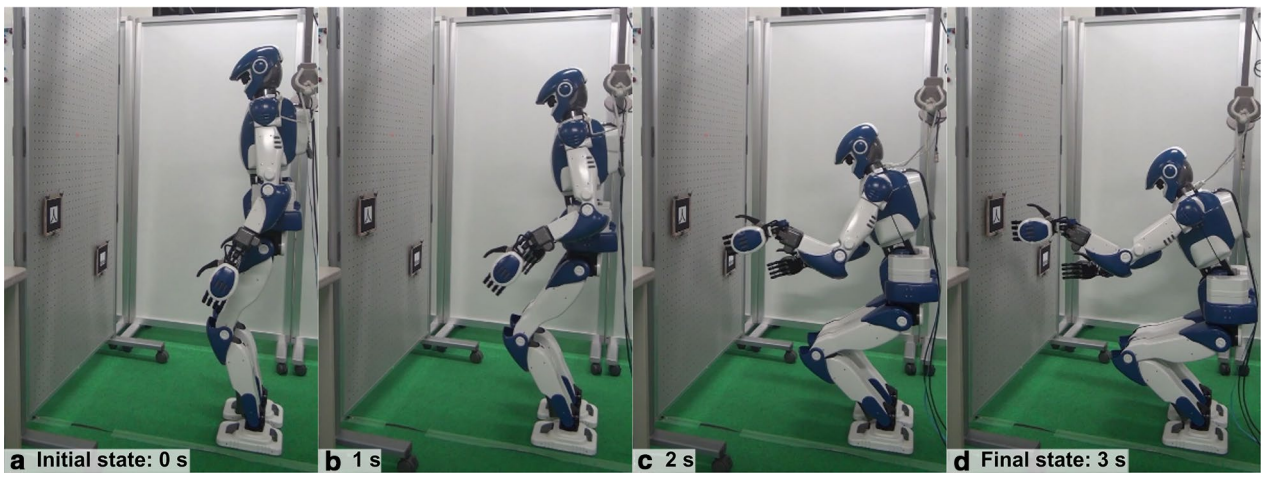

Fig. 14 The proposed method generates whole-body motion to make the HRP-4 reach with both hands the target positions indicated with AR markers. By managing the CoM in the center of the foot's soles, the robot can keep its balance while hunkering down (**a** initial state, **b** and **c** intermediate states, and **d** final state)

in the last part of the motion. In the case of our proposed method, we should add a threshold of a horizontal CoM to avoid falling down. After approaching this threshold, the robot should move slowly. Another option is including a locomotion for approaching the whole body out of this threshold to our proposed method.

Conclusions

In this paper, we proposed a method for whole-body motion generation for humanoid robots in household environments to reach objects with their hands in real time. To reduce the calculation time of motion generation, we estimated the torso posture using SVR before calculating the joint angles. Using the estimated torso posture, we can separate the IK for the whole body into simpler, independent IK for the arms and the lower body. We solve the IK of both arms analytically and solve the IK of the lower body numerically. To sequentially reach multiple targets for the hands, we built a target position generator for the CoM considering the reachability of both arms.

In terms of calculation time, the proposed method is faster than the conventional Jacobian-based numerical method, while achieving better accuracy. To verify the effectiveness of the proposed method, we implemented it on a human-sized humanoid robot, the HRP-4. In experiments, we successfully generated reaching motions for lower positions, which included hunkering motion while keeping the balance. In these experiments, it took 3 s to both generate and execute the target motion. We

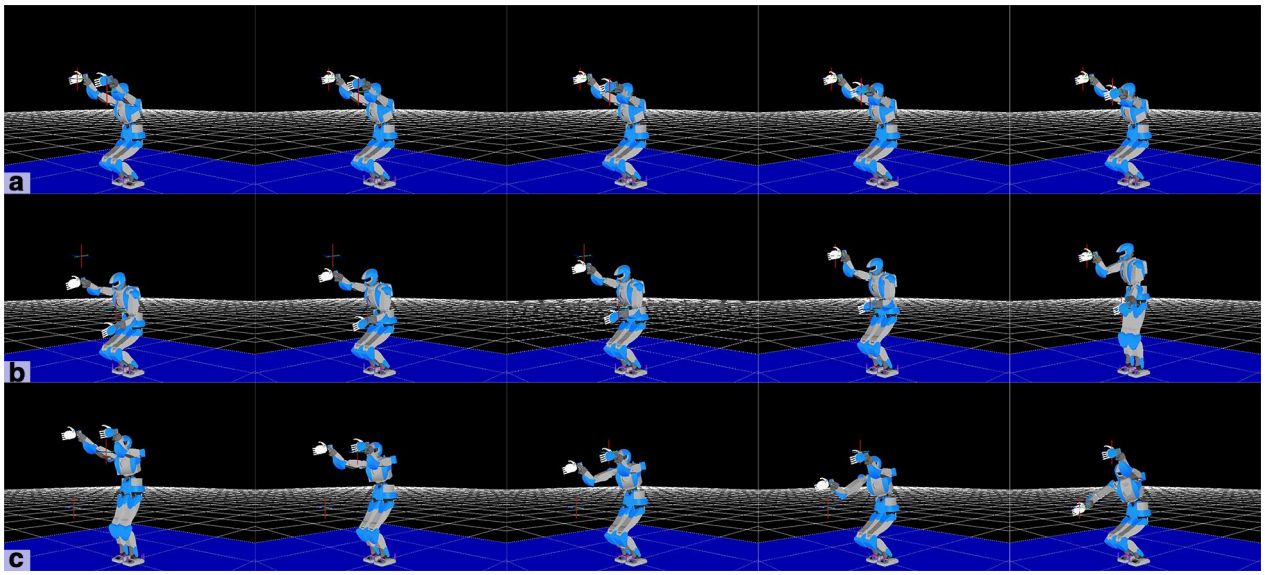

Fig. 15 The simulation result when approaching random target positions. These movements include whole-body motion in the vertical direction and the arms' motion in the vertical and diagonal direction (**a**, **b**, and **c** show the movement with different target positions)

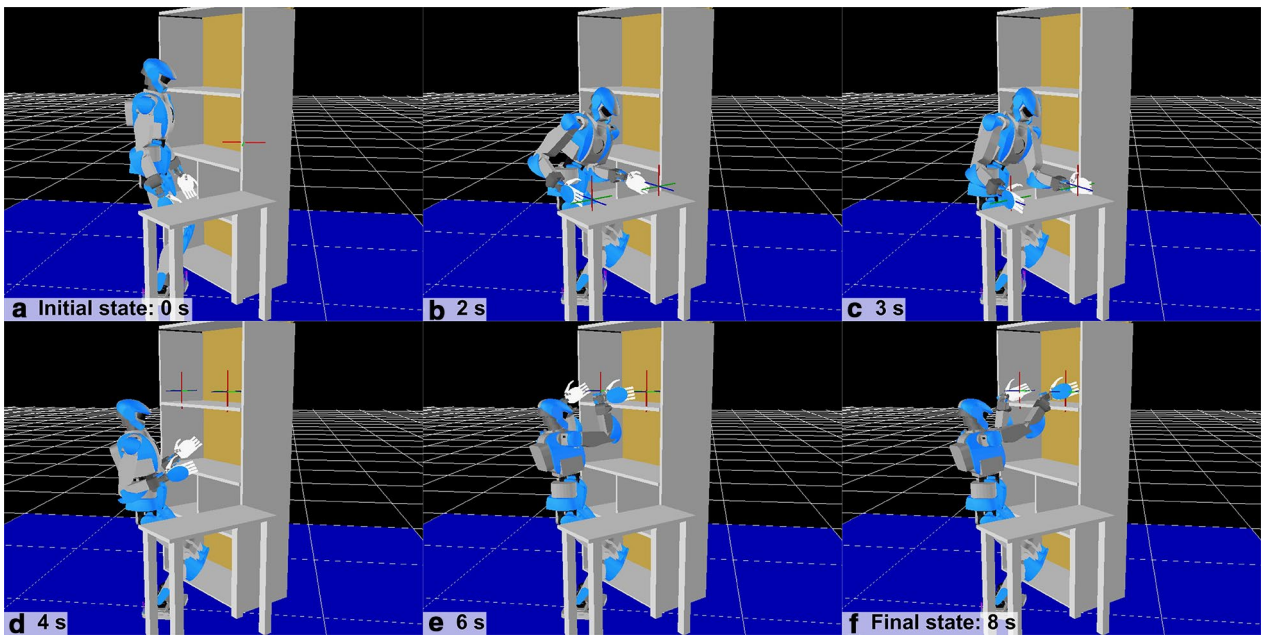

Fig. 16 Simulation results when approaching target positions with both hands in a living environment. There is a table and a shelf in the dynamics simulation environment. To avoid collisions, we set waypoints when approaching the target positions of each object (parts (**b**) and (**e**) in this figure). First, the HRP-4 approaches the target objects on the table. Then, the HRP-4 moves to the target objects on the shelf (**a** initial state, **c** and **d** intermediate states, and **f** final state)

also showed the applicability of the proposed method to a step-by-step manipulation by sequentially inputting random targets. In experiments using a dynamics simulation, HRP-4 could approach the target positions smoothly. Finally, we also applied the proposed method to a kitchen environment, by generating a whole-body motion to approach a table and an upper shelf.

In this paper, we only generated motions when the humanoid's feet were fixed. Since the proposed lower-body controller manages the configuration of the swing leg, the proposed method has the potential to expand the reachable area by stepping. Further, in the posture estimation, we employed SVR as the regression method to simplify implementation, but we could investigate using other regression methods such as random forest regression and neural networks.

Though we did not test our method with the inertial changes present when grasping an object, we are planning to do it as future work. The proposed method should be able to adapt to these changes if the physical properties of the grasped object are known *a priori*.

Additional files

Additional file 1. Whole-body motion to reach with both hands two AR markers using the proposed method.

Additional file 2. Sequential whole-body control with random inputs.

Additional file 3. Whole-body motion to reach with both hands in a living environment.

Abbreviations

IK: inverse kinematics; DoF: degree(s) of freedom; CoM: center of mass; SVR: support vector regression; ZMP: zero moment point.

Authors' contributions

ST, and JT developed the concept and approach, implementing the structure of the proposed method, data acquisition, and drafting of the manuscript. YH implemented the arm controller using an analytical method. GAGR, JT and TO carried out the conceptual supervising. All authors read and approved the final manuscript.

Authors' information

Satoki Tsuichihara received his M.E. degree from the Nara Institute of Science and Technology, Japan, in 2013. His research interests include humanoid robotics, whole-body motion generation, vision-based manipulation, and service robots. (satoki-t@is.naist.jp).

Yuya Hakamata received his M.E. degree from the Nara Institute of Science and Technology, Japan, in 2016. His research interests include humanoid robotics, whole-body motion generation, and contact force control. (hakamata.yuya.ht9@is.naist.jp).

Gustavo Alfonso Garcia Ricardez received his Ph.D. degree from the Nara Institute of Science and Technology, Japan, in 2016. He is currently an Assistant Professor in the Robotics Laboratory of the Division of Information Science at the Nara Institute of Science and Technology. His research interests include human-safe and efficient robot control, human-robot interaction, and robotics competitions. (garcia-g@is.naist.jp).

Jun Takamatsu received his Ph.D. degree in Computer Science from The University of Tokyo, Japan, in 2004. From 2004 to 2008, he was with the Institute of Industrial Science, University of Tokyo. In 2007, he was with Microsoft Research Asia, as visiting researcher. In 2008, he joined the Nara Institute of Science and Technology, Japan, as Associate Professor. His research interests include task and motion planning, learning from observation, feasible motion analysis, 3D-shape modeling and analysis, and physics-based vision. (j-taka@is.naist.jp).

Tsukasa Ogasawara received his Ph.D. degree from The University of Tokyo, Japan, in 1983. From 1983 to 1998, he was with the Electrotechnical Laboratory, Ministry of International Trade and Industry, Japan. From 1993 to 1994, he was with the Institute for Real-Time Computer Systems and Robotics, University of Karlsruhe, Germany, as a Humboldt Research Fellow. He joined the Nara Institute of Science and Technology, Nara, Japan, in 1998, where he is currently a Professor in the Division of Information Science. He is a Vice President of NAIST and Dean of the Graduate School of Information Science. His research interests include human-robot interaction, dexterous manipulation, human modeling and biologically inspired robotics. (ogasawar@is.naist.jp).

Acknowledgements

This work was supported by JSPS KAKENHI Grant Number JP16K12502.

Competing interests

The authors declare that they have no competing interests.

Availability of data and materials

The data supporting the conclusions of this article are included within the article.

Consent for publication

No personal data.

Ethics approval and consent to participate

No experiments with humans.

Publisher's Note

Received: 23 June 2017 Accepted: 16 April 2018
Published online: 27 April 2018

References

1. Nozawa S, Murooka M, Noda S, Okada K, Inaba M. (2013) Description and execution of humanoid's object manipulation based on object-environment-robot contact states. In: IEEE international conference on intelligent robots and systems (IROS), pp 2608–2615
2. Shigemi S, Kawaguchi Y, Yoshiike T (2006) Development of new ASIMO. Honda R D Tech Rev 18(1):38–44
3. Okada K, Kojima M, Sagawa Y, Ichino T, Sato K, Inaba M. (2006) Vision based behavior verification system of humanoid robot for daily environment tasks. In: IEEE-RAS international conference on humanoid robots (Humanoids). pp 7–12
4. Kajita S, Hirohisa H, Kensuke H, Kazuhito Y. (2014) Introduction to humanoid robotics. pp 12–17
5. Gates B (2007) A robot in every home. Sci Am 296(1):58–65
6. Cognetti M, Fioretti V, Oriolo G. (2016) Whole-body planning for humanoids along deformable tasks. In: IEEE international conference on robotics and automation (ICRA). pp 1615–1620
7. Kaneko K, Kanehiro F, Morisawa M, Akachi K, Miyamori G, Hayashi A, Kanehira N. (2011) Humanoid robot HRP-4-Humanoid robotics platform with lightweight and slim body. In: IEEE/RSJ international conference on intelligent robots and systems (IROS). pp 4400–4407
8. Tsuichihara S, Hakamata Y, Garcia Ricardez GA, Takamatsu J, Ogasawara T. (2016) Accelerating whole-body motion generation using regression of the torso posture of a humanoid robot. In: IEEE-RAS international conference on humanoid robots (Humanoids). pp 16–21
9. Henze B, Roa MA, Ott C (2016) Passivity-based whole-body balancing for torque-controlled humanoid robots in multi-contact scenarios. Int J Robot Res 35(12):1522–1543
10. Nozawa S, Kanazawa M, Kakiuchi Y, Okada K, Yoshiike T, Inaba M. (2016) Three-dimensional humanoid motion planning using COM feasible region and its application to ladder climbing tasks. In: IEEE-RAS international conference on humanoid robots (Humanoids). pp 49–56
11. Vannoy J, Xiao J (2008) Real-time adaptive motion planning (RAMP) of mobile manipulators in dynamic environments with unforeseen changes. IEEE Trans Robot 24(5):1199–1212
12. Park C, Pan J, Manocha D. (2008) ITOMP: Incremental trajectory optimization for real-time replanning in dynamic environments. In: Proceedings of

the international conference on automated planning and scheduling. pp 207–215
13. Otte M, Frazzoli E. (2015) RRT X: Real-time motion planning / replanning for environments with unpredictable obstacles. In: Algorithmic foundations of robotics XI: selected contributions of the eleventh international workshop on the algorithmic foundations of robotics. pp 461–478
14. Fok C-L, Johnson G, Sentis L, Mok A, Yamokoski JD (2015) ControlIt! —a software framework for whole-body operational space control. Int J Humanoid Robot 13(01):1550040
15. Nishiwaki K, Kuga M, Kagami S, Inaba M, Inoue H (2005) Whole-body cooperative balanced motion generation for reaching. Int J Humanoid Robot 2(4):437–457
16. Yamane K, Nakamura Y (2003) Natural motion animation through constraining and deconstraining at will. IEEE Trans Vis Comput Graph 9:352–360
17. Ferrari P, Cognetti M, Oriolo G. (2017) Humanoid whole-body planning for loco-manipulation tasks. In: IEEE international conference on robotics and automation (ICRA). pp 4741–4746
18. Cognetti M, Mohammadi P, Oriolo G. (2015) Whole-body motion planning for humanoids based on CoM movement primitives. In: IEEE-RAS international conference on humanoid robots (Humanoids). pp 1090–1095
19. Khatib O (1986) Real-time obstacle avoidance for manipulators and mobile robots. Int J Robot Res 5(1):90–98
20. Zucker M, Ratliff N, Dragan AD, Pivtoraiko M, Klingensmith M, Dellin C, Bagnell JA, Srinivasa S (2013) CHOMP: covariant Hamiltonian optimization for motion planning. Int J Robot Res 32(9–10):1164–1193
21. Yang P-C, Sasaki K, Suzuki K, Kase K, Sugano S, Ogata T (2016) Repeatable folding task by humanoid robot worker using deep learning. IEEE Robot Autom Lett (RA-L) 2(2):397–403
22. Smola AJ, Schölkopf B (2004) A tutorial on support vector regression. Stat Comput 14(3):199–222
23. Sugihara T (2011) Solvability-unconcerned inverse kinematics by the Levenberg-Marquardt method. IEEE Trans Robot 27(5):984–991
24. Murray RM, Sastry SS, Zexiang L (1994) A mathematical introduction to robotic manipulation, 1st edn. CRC Press Inc, Boca Raton
25. Hirokazu K, Billinghurst M. (1999) Marker tracking and HMD calibration for a video-based augmented reality conferencing system. In: IEEE and ACM international workshop on augmented reality (IWAR). IWAR '99. pp 85–94
26. Kanehiro F, Hirukawa H, Kajita S (2004) OpenHRP: open architecture humanoid robotics platform. Int J Robot Res 123(2):155–165