Online transfer of human motion to humanoids

Behzad Dariush, Michael Gienger, Arjun Arumbakkam, Youding Zhu, Bing Jian, Kikuo Fujimura, Christian Goerick

2009

International Journal of Humanoid Robotics
© World Scientific Publishing Company

ONLINE TRANSFER OF HUMAN MOTION TO HUMANOIDS

BEHZAD DARIUSH
Honda Research Institute, USA, Mountain View, CA, 94041, USA
dariush@honda-ri.com

MICHAEL GIENGER
Honda Research Institute, Europe, Carl-Legien-Strasse 30 63073, Offenbach/Main, Germany
michael.gienger@honda-ri.de

ARJUN ARUMBAKKAM
Honda Research Institute, USA, Mountain View, CA, 94041, USA
aarumbakkam@honda-ri.com

YOUDING ZHU
The Ohio State University, Columbus, OH, USA
zhu.81@osu.edu

BING JIAN
Department of Computer and Information Science and Engineering, University of Florida, Gainesville, FL 32611 bjian@cise.ufl.edu

KIKUO FUJIMURA
Honda Research Institute, USA, Mountain View, CA, 94041, USA
kfujimura@honda-ri.com

CHRISTIAN GOERICK
Honda Research Institute, Europe, Carl-Legien-Strasse 30 63073, Offenbach/Main, Germany
christian.goerick@honda-ri.de

Received Day Month Year
Revised Day Month Year
Accepted Day Month Year

Transferring motion from a human demonstrator to a humanoid robot is an important step toward developing robots that are easily programmable and that can replicate or learn from observed human motion. The so called motion retargeting problem has been well studied and several off-line solutions exist based on optimization approaches that rely on pre-recorded human motion data collected from a marker-based motion capture system. From the perspective of human robot interaction, there is a growing interest in online motion transfer, particularly without using markers. Such requirements have placed stringent demands on retargeting algorithms and limited the potential use of off-line and pre-recorded methods. To address these limitations, we present an online task space control theoretic retargeting formulation to generate robot joint motions that adhere to the robot's joint limit constraints, self-collision constraints, and balance constraints. The inputs to the proposed method include low dimensional normalized human motion descriptors, detected and tracked using a vision based key-point detection and tracking algorithm. The proposed vision algorithm does not rely on markers placed on anatomical landmarks, nor does it require special instrumentation or calibration. The current implementation requires a depth image sequence, which is collected from a single time of flight imaging device. The feasibility of the proposed approach is shown by means of online experimental results on the Honda humanoid robot - Asimo.

Keywords: Retargeting, Imitation Learning, Task Space Control

1. Introduction

Learning from human demonstration, also referred to as imitation learning, has become an important topic of research in robotics with applications spanning across many disciplines such as robot motion control, human-robot interaction, and machine learning. Imitation learning promises to simplify the process of programming complex humanoid robot motions by replacing the time-consuming manual programming of a robot by an automatic programming process, solely driven by showing the robot the task by an expert teacher [23] [19]. Examples from a human demonstrator also provide a powerful mechanism for reducing the complexity of search spaces in learning algorithms by either starting the search from the observed locally optimal solutions [10], or by eliminating from the search space what are known as infeasible solutions.

A long standing trend in learning from demonstration methods has been to approach the problem from the standpoint of motion replication, although recent work inspired by this rational is not just about observing and replicating the motion, but rather about understanding the goals of a given action. Indeed, many aspects of imitation are goal-directed, that is, actions are meant to fulfill a specific purpose and convey the intention of the teacher [2]. Nevertheless, one can argue that an important pre-requisite step in learning from imitation involves mimicry, or the reproduction of the actions of the human demonstrator.

Mimicry can be viewed as a sophisticated transfer of the observed human motion to the robot, a procedure referred to as motion retargeting by the computer graphics community [9,14,28,27,7]. The representation of motion by descriptors which capture the essence of motion or encode meaningful information about the task is an important research problem. Typically, motion descriptors are simply described by either joint space or task (Cartesian) space coordinates. Task space methods offer an advantage in that the large number of mechanical degrees of freedom involved in the execution of movement tasks by humanoids can be represented by lower dimensional descriptors. These descriptors may be referred to as task descriptors because they are used to describe motion by higher level task variables which may be encoded to convey the intention of the human performer. A task oriented approach is also compatible with current views in motor learning that suggest that the central nervous system organizes or simplifies the control of large degrees of freedom during motion execution and motor learning phases. That is, the controlled operation of the neuromuscular system with an exorbitant number of degrees of freedom requires a reduction of mechanical redundancy, achieved by reducing the number of degrees of freedom [3].

Regardless of how the motion is represented, by task variables or joint variables, a suitable retargeting algorithm must deal with complex kinematic constraints due to differences in topology, anthropometry, joint limits, and degrees of freedom between the human demonstrator and the robot. Similar problems have been widely studied and partially addressed, particularly for transferring human motion to an animated character. The problem is often formulated and solved as a constrained non-linear optimization problem, where the objective is to minimize the error between the human motion and the target motion, subject to the kinematic constraints [29,18]. Such approaches are often performed off-line, in static environments, using pre-recorded human motion obtained from marker based motion capture systems [20,22]. The resulting motion is then used as the reference trajectory to be executed by the robot's motion controller.

Off-line methods using prerecorded motions do not account for a dynamically changing environment and have no provision for sensory feedback from the robot's current state to the motion retargetter. Therefore, there is a lack of robustness to uncertainties in the environment. Assuming the environment is static, the edited motion is likely to be admissible by the robot's structure and can to a certain degree be executed by the robot's control scheme during run-time. Balance constraints [13], and obstacle avoidance [24] are sometimes considered. To our knowledge, self collisions constraints have not been explicitly considered as part of a motion retargeting procedure, although recent work demonstrated a control formulation for a real time self collision avoidance on a humanoid robot [26].

In many human-robot interaction applications, the requirements of interactivity in dynamically changing environments as well as simplicity in sensing and instrumentation, have placed stringent demands on the retargeting procedure. One such application requiring online and interactive performance involves imitative social interaction of robots with children with learning disabilities, such as autism [21]. These requirements include capturing human motions unobtrusively, without instrumenting them with markers. Also, human motion capture with multiple cameras in a special environment may neither be feasible nor practical. The imaging modality must be simple and the underlying retargeting mechanism must cope with this.

In this paper, we present an online, Cartesian space control theoretic retargeting formulation to generate robot joint motions that adhere to the robot's joint limit constraints, self-collision constraints, and balance constraints. The inputs to the proposed method include low dimensional normalized human task descriptors, detected and tracked using a vision based key-point detection and tracking algorithm which is described in detail in our previous work [31]. The visual processing algorithm does not rely on markers placed on anatomical landmarks, nor does it require special instrumentation or calibration. The current implementation requires a depth image sequence, which is collected from a single time of flight imaging device. We present online experimental results of the entire pipeline on the Honda humanoid robot - Asimo.

The paper is organized as follows. An overview of the entire motion retargeting pipeline, from image acquisition to humanoid robot motion control, is given in Section 2. Section 3 describes an overview of the visual procession module used to detect key feature points on the human body. Using these detected key-points, we describe the proposed task space retargeting framework in Section 4. The retargeting framework includes the description of the Cartesian space tracking (Section 4.2), task management strategies (Section 4.3), joint limit avoidance (Section 4.4), joint velocity limiting (Section 4.5), self collision avoidance (Section 4.6), and balance control (Section 4.7). Experimental results of the online retargeting using a markerless, single depth camera system, are reported in Section 6. The paper is summarized in Section 7.

2. Overview of the entire pipeline

Figure 1 illustrates an overview of the proposed online motion retargeting framework. The first step involves visual detection and tracking of a set of 3D anatomical landmarks (or key-points) in the upper-body from image observations using a time of flight depth imaging device [1]. The detected key-points, registered to a human model, correspond to 3D position vectors at the waist joint, two shoulder joints, two elbow joints, two wrist joints, and the head center (Figure 2). The output of the key-point detection module is represented by the vector p_d, where the subscript d denotes detected key-point. For completeness, the theoretical formulation of the proposed retargeting algorithm considers not only the position of key-points as input, but also the orientation variables if such information becomes available. Orientation information such as head pose provide important visual cues in imitation learning; however, orientation is not directly observable based on our current vision module and is not considered in our experimental results. If such information becomes available, each detected orientation descriptor may be described by a rotation matrix R_d.

The detected key-points are subsequently low pass filtered and normalized (limb lengths re-scaled) to our humanoid robot model, Asimo, which has different dimensions, physical parameters, geometry, and degrees of freedom than the human model. Furthermore, the filtered and scaled key-points are up-sampled to a higher rate (100 HZ) to achieve numerical stability and good tracking within the retargeting module. The resulting vector, denoted by the vector p_r represents the reference motion of a key-point mapped to the robot. In this paper, we will refer to these robot motion variables as task descriptors because they correspond to Cartesian space (or task space) positions and orientations in our proposed task space retargeting framework. If available, a reference orientation descriptor may be denoted by the rotation matrix R_r. Taking the reference task descriptors as input, the retargeting module outputs kinematically constrained robot joint variables which are issued as commands to the robot.

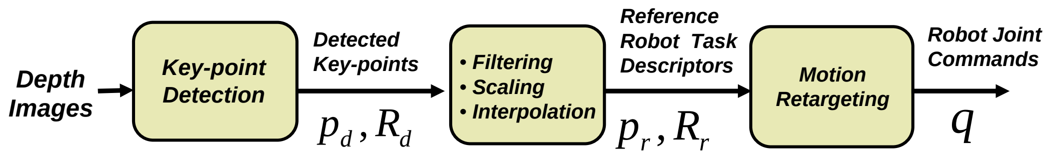


Fig. 1. System diagram of the entire pipeline.

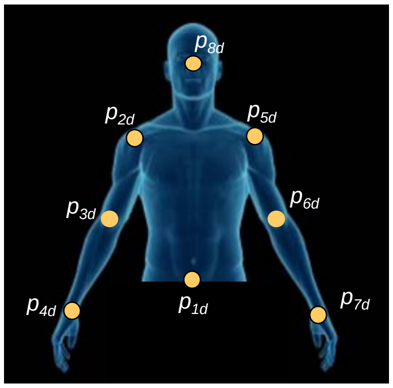


Fig. 2. Key feature points (key-points) representing position descriptors used in the experiments.

3. Key-point Detection

We use depth image streams to extract the key feature points (key-points) in the upper body. The depth images, obtainable by using active or passive stereo, or time-of-flight sensors, provides a 3D point cloud of the scene, including the human body. Our experimental results are based on a single 3D time of flight depth camera (Swiss Ranger-SR 3000 [1]) which captures depth and intensity images at approximately 15 frames per second. An advantage of TOF cameras is their portability, relatively good depth resolution as compared with passive stereo systems, and very little or no required calibration. In addition, the nature of images captured by such devices facilitates segmentation of the human from the background clutter.

The key-point detector is based on a Bayesian framework which utilizes both spatial context and temporal information to robustly detect and track limbs and identify key-points in the presence of intermittent self-occlusions. Details of the key-point detection algorithm are described in [31]. We illustrate the key-point detection

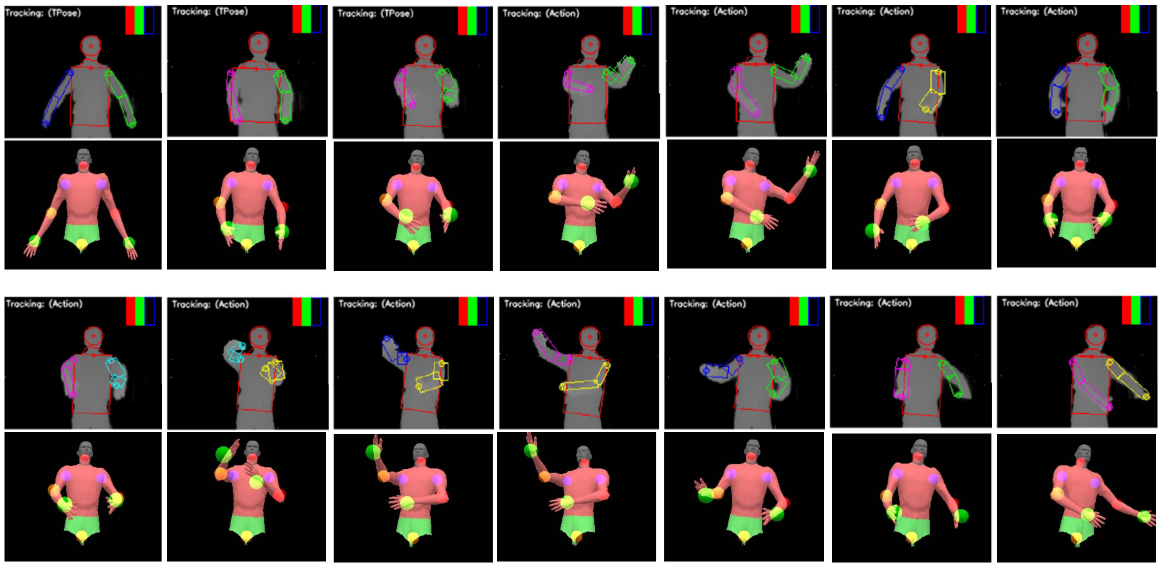


Fig. 3. Taiji dance (upper-body). Rows 1 and 3: depth image sequence with the detected limb templates superimposes. Rows 2 and 4: corresponding reconstructed pose.

and human body pose estimation results for a Taiji dance sequence as shown in Figure 3. The figure illustrates snapshots of the depth image sequence and the corresponding reconstructed human pose. In particular, Rows 1 and 3 illustrate body limb templates which are registered to the 3D depth image. Key-points are detected based on the location of these templates. The detected key-points are then used to reconstruct the 3D human model pose (shown in Rows 2 and 4). The detected key-points are denoted by the small spheres superimposed on the 3D human model.

The position of each detected key-point is represented by the vector $p_{d_i}, i \in \{1, \ldots, 8\}$ and described in the base frame corresponding to the waist joint coordinate system. We establish correspondence between the detected descriptors (or key-points) on the human model and reference descriptors on the humanoid model through simple scaling operations for each link. Subsequently, each detected key-point is normalized to the robot dimensions, filtered using a 2nd order Butterworth low pass filter, and interpolated using Cardinal splines interpolation. The resulting position vector, denoted by p_{r_i}, represents the reference task descriptors on the robot model.

4. Retargeting

The proposed retargeting algorithm can be described as a local constrained optimization problem. The objective is to estimate the robot joint commands that minimize the tracking error between the reference and predicted task descriptors subject to kinematic and balance constraints. Although the detected key-points from our current vision algorithm describe the 3D position descriptors corresponding to anatomical landmarks, for generality, we formulate our retargeting algorithm such that orientation descriptors can also be incorporated if such information becomes available. Therefore, our formulation below considers task descriptors which operate the full six dimensional task space, three for position and three for orientation. If an orientation descriptor is available, it is described by a reference rotation matrix R_r. Both the position and orientation quantities are described in the base frame corresponding to the waist joint coordinate system.

4.1. *Differential Kinematics*

Let n represent the number of robot upper body joint variables. Let the vector $q = [q_1, \cdots, q_n]^T$ describe the degrees of freedom which fully characterize the configuration space, or joint space, of the upper-body humanoid robot. For generality, suppose there are k task variables to control in our humanoid model. Let i $(i = 1 \cdots k)$ be the index of the spatial velocity vector $\dot{x}_i$ corresponding to the i_{th} task descriptor. The associated Jacobian is given by $J_i = \frac{\partial x_i}{\partial q}$.The mapping between the joint space velocities and task space velocities is obtained by considering the differential kinematics relating the two spaces,

$$\dot{x}_i = J_i(q)\,\dot{q}. \tag{1}$$

The spatial velocity vector is defined by

$$\dot{x}_i = \begin{bmatrix}\omega_i \ \dot{p}_i\end{bmatrix}^T, \tag{2}$$

where ω_i and $\dot{p}_i$ are vectors corresponding to the angular velocity of the task frame and the linear velocity of the task position referenced to the base frame, respectively. If the Euler angles Θ are available, the angular velocity can be computed from,

$$\omega_i = \mathrm{H}_i(\Theta_i)\ \dot{\Theta}_i, \tag{3}$$

where the transformation matrix H_i depends on the particular set of Euler angle sequence considered. Alternatively, the angular velocities can be directly extracted from the rotation matrix through

$$\tilde{\omega}_i = \dot{R}_i R_i^T, \tag{4}$$

where $\tilde{\omega}_i$ corresponds to a 3×3 the skew symmetric matrix containing the elements of the angular velocity vector. To augment all task variables, we form an augmented spatial velocity vector $\dot{x}$ and an augmented Jacobian matrix J as follows,

$$\dot{x} = \begin{bmatrix}\dot{x}_1^T & \cdots & \dot{x}_i^T & \cdots & \dot{x}_k^T\end{bmatrix}^T, \tag{5}$$

$$J = \begin{bmatrix}J_1^T & \cdots & J_i^T & \cdots & J_k^T\end{bmatrix}^T. \tag{6}$$

The Jacobian matrix may be decomposed to its rotational and translational components, denoted by J_o and J_p, respectively.

$$J = \begin{bmatrix} J_o \\ J_p \end{bmatrix}. \tag{7}$$

If for example, only a position descriptor p_i is observable, then the parameters in Equation 1 can be modified to $\dot{x}_i = \dot{p}_i$, and $J_i = J_{p_i}$.

4.2. *Cartesian tracking control*

We wish to create a control policy that produces the robot joint commands (q) such that the Cartesian error between the predicted robot task descriptors and the normalized human task descriptors are minimized. The tracking performance is very much subject to the robot's kinematic constraints, as well as the execution of multiple and often conflicting task descriptor requirements. Our formulation of such a constrained optimization is based on extensions of a Cartesian space kinematic control method known as closed loop inverse kinematics (CLIK). In this section, we provide an overview of CLIK in the context of motion retargeting. Subsequently, we incorporate kinematic constraints to the CLIK formulation.

Let the reference task descriptors in the augmented space be described by,

$$\dot{x}_r = \begin{bmatrix} \dot{x}_{r_1}^T & \cdots & \dot{x}_{r_i}^T & \cdots & \dot{x}_{r_k}^T \end{bmatrix}^T. \tag{8}$$

The joint velocities may be computed by inverting Equation (1) and adding a feedback error term to correct for numerical drift.

$$\dot{q} = J^*(\dot{x}_r + K\ e), \tag{9}$$

where J^* denotes the regularized right pseudo-inverse of J weighted by the positive definite matrix W,

$$J^* = W^{-1}J^T\ (JW^{-1}J^T + \lambda^2 I)^{-1}. \tag{10}$$

The parameter $\lambda > 0$ is a damping term, and I is an identity matrix. The vector $\dot{x}_{r_i} = [\omega_{r_i} \quad \dot{p}_{r_i}]^T$ corresponds to the differential kinematics of the reference motion and ω_{r_i} may be calculated based on Equation 3. The rate of convergence of the error for the i_{th} descriptor is controlled by K_i, a diagonal 6×6 positive definite gain matrix. The vector e is the concatenation of the individual error terms $e_i = [e_{o_i}\ e_{p_i}]^T$, where$(e_{p_i})$ and (e_{o_i}) are the position and orientation error vectors between the reference and computed task descriptors. The position error is simply defined as $e_{p_i} = p_{r_i} - p_i$, where p_{r_i} and p_i correspond to the reference and computed task descriptor positions, respectively. The computation of the orientation error is more complex. Orientation is typically described by three Euler angles, denoted here by the vector Θ_{r_i}. The Euler angles can also be calculated if the reference rotation matrix is known. Let $R_{r_i} = [n_r \quad s_r \quad a_r]$ and $R_i = [n \quad s \quad a]$ correspond to the reference and computed unit vector triple representation of the task descriptor frame orientation, respectively. A functional expression of the orientation error in terms of an angle and axis error is given by [15],

$$e_o = \frac{1}{2}(n \times n_r + s \times s_r + a \times a_r). \tag{11}$$

Describing the orientation error using Equation 11 is convenient because there is no need to explicitly extract Euler angles from the rotation matrix: a procedure which can be problematic due to the non-unique and sometimes discontinuous solutions.

The block-diagram of the tracking control based on the CLIK algorithm is illustrated in Figure 4. This algorithm is used in order to arrive at a configuration which tracks the position and orientation of the observed motion descriptors. Equation 9 is used in order to arrive at a configuration which tracks the position and orientation of the observed motion descriptors.

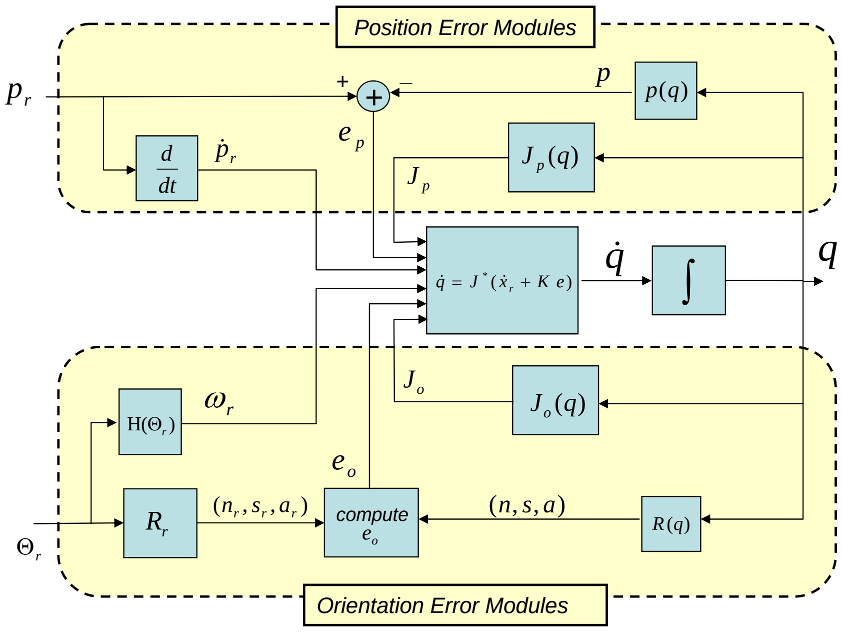


Fig. 4. Detailed system diagram of a first order closed loop inverse kinematics (CLIK) tracking control with partitioned position and orientation control modules.

4.3. *Managing multiple task descriptors*

An important question yet to be addressed is the how to manage multiple and often conflicting task requirements. Task management strategies are important from a sensing standpoint to ensure that key-points detected with a higher confidence are assigned a higher priority for tracking. For instance, detecting elbow locations are often more difficult than other anatomical landmarks associated with end-effectors, such as the hands. In such cases, the relative confidence values in the detected key-points provide insight on how well a particular key-point should be tracked.

In robotics, a more conventional use of task management relates to the actual task execution by the robot according to the importance of achieving particular task relative to other tasks. For example, to replicate a human demonstrator grasping a cup of water, faithfully tracking the grasping hand posture is more important than tracking other tasks. In this section, we describe two methods for managing multiple task descriptors in our retargeting framework. These methods are referred to as *task augmentation* and *task prioritization.*

4.3.1. *Task Augmentation*

The formulation presented in Section 4.2 is an example of task descriptor augmentation, referring to the concatenation of individual spatial velocities and the associated Jacobian matrices and feedback gain matrices. An effective method to manage the descriptors is to modulate the tracking error rate. The error rate for each element of a task descriptor can be controlled by the augmented feedback gain matrix K, which represents a diagonal matrix in the augmented space. The trajectory tracking error convergence rate depends on the eigenvalues of the feedback gain matrix in Equation (9); the larger the eigenvalues, the faster the convergence. In practice, such systems are implemented as a discrete time approximation of the continuous time system; therefore, it is reasonable to predict that an upper bound exists on the eigenvalues, depending on the sampling time. A particular task descriptor or its individual components can be more tightly tracked by increasing the eigenvalues of K associated with that direction. By modulating the elements of K, we can effectively encode the relative level of confidence we have in our observations. Measurements with higher confidence will be assigned higher feedback gain values. Likewise, the relative importance of tasks for robot execution can be modulated by choosing the proper feedback gains. In general, task augmentation is suitable when we are more interested in capturing the essence of the entire human motion, rather than precisely executing a specific task.

We tested the task augmentation strategy on simulated human reaching motion obtained from the Carnegie Mellon University (CMU) human motion data base [5]. The raw human motion data in the CMU data-set is represented in C3D file format, from which we extracted eight 3D upper body Cartesian positions corresponding to the key-points in Figure 2. From the possible 8 descriptors, we selected only 3 motion descriptors (left hand, right hand, and waist) to show the tracking results using task augmentation. Figure 5 illustrates the effect of varying the feedback gain (K) on the tracking error for three different sets of simulations. For each simulation trial, we recorded the root mean squared error (RMSE) between the observed and predicted motion descriptors using three different values for the feedback gain. In each simulation trial, we assigned a feedback gain of $K = 150\ I_3$ for one task descriptor and $K = 50\ I_3$ for the remaining two, where I_3 represents a 3×3 identity matrix. The plot illustrates that for a given task descriptor, the tracking error across the different simulation trials (intra-trial error) can be reduced by increasing the feedback gain associated with that descriptor. However, modulation of the feedback gain in the task augmentation framework cannot guarantee that a particular task will be precisely executed with certainty. In theory, as will be described in the next section, task prioritization can precisely execute the highest priority task if there are no algorithmic or task singularities. In practice, however, such requirements are difficult to realize in our retargeting application.

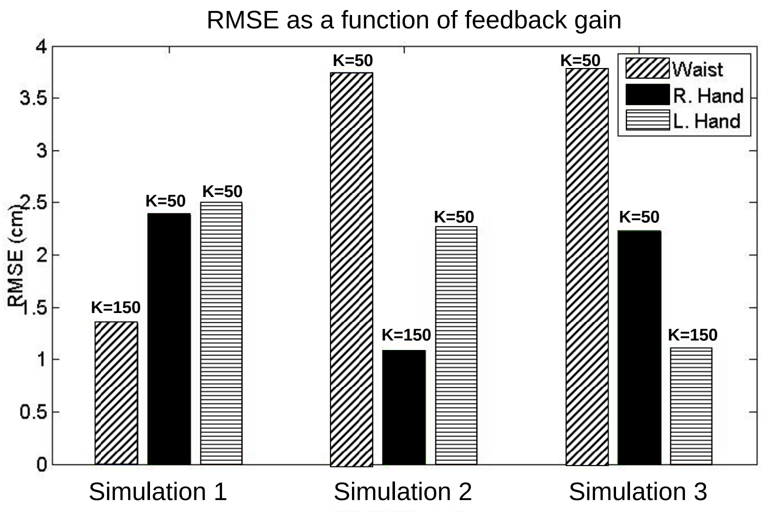


Fig. 5. Illustration of simulated RMS tracking error for a reaching motion obtained from the CMU data-set.

4.3.2. *Task Prioritization*

Multiple tasks can be managed by prioritizing their execution, particularly if there is a high degree of redundancy. Recursive methods which handle an arbitrary number of prioritized task descriptors have been previously described [25]. We develop an alternative algorithm here.

Suppose we wish to assign a priority on the execution of a particular task descriptor or groups of task descriptors. For example, suppose we form $i = 1 \cdots k$ sub-groups of tasks (or subtasks). Here, the index i is associated with the i_{th} subtask, where a sub-task may be associated with a single task descriptor, or a group of task descriptors which are concatenated using task augmentation. The total task description may be composed of k subtasks with the order of priority. Let m_i be the dimension of each subtask. For execution of tasks in group i, the differential kinematic relationship between the joint velocity $\dot{q} \in \Re^n$ and the Cartesian variable $\dot{x}_i \in \Re^{m_i}$ is expressed by,

$$\dot{q} = J_i^+(q)\dot{x}_i. \tag{12}$$

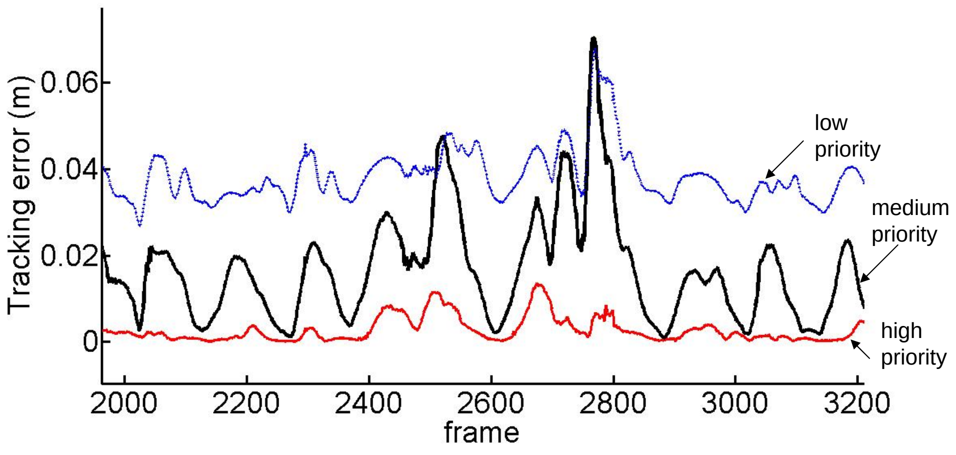
where J_i is the Jacobian matrix of the i_{th} subtask descriptor and J^+ is typically defined as right pseudo-inverse of J, given by $J^+ = J^T(JJ^T)^{-1}$. A solution for a prioritized two task problem has been outlined in [16]. A prioritized solution can be extended to more than two tasks following the same procedure, as given in Algorithm 1.

Algorithm 1: General solution for multiple tasks with the order of priority

```
Input: $J_i \in \Re^{m_i \times n}$, $\dot{x}_i \in \Re^{m_i}$, $i = 1, \ldots, k$,   where $k$ is the number of
       subtasks
Output: $\dot{q}$
begin
    $N_0 = I$
    for $i \leftarrow 1$ to $k$ do
        $v_i = \dot{x}_i$
        $\hat{J}_i = J_i N_{i-1}$
        $\hat{v}_i = v_i - J_i \sum_{j=1}^{i-1}(\hat{J}_j^+ \hat{v}_j)$
        $N_i = N_{i-1}(I - \hat{J}_i^+ \hat{J}_i)$
    $\dot{q} = \sum_{i=1}^{k}(\hat{J}_i^+ \hat{v}_i) + N_k z$   where $z \in \Re^n$ is an arbitrary vector
end
```

A key fact in Algorithm 1 is that all N_i's are orthogonal projectors, which is used to derive the identity $N_{i-1}\hat{J}_i^+ = \hat{J}_i^+$. Note that $\{N_i\}$ form a sequence of orthogonal projectors with decreasing ranks. The final solution space can be considered as the intersection of the k solution subspaces determined by the k subtasks. Let J be the matrix of size $\sum m_i \times n$ obtained by stacking J_i's, the exact solution space is not empty if and only if J is of full row rank, i.e. $rank(J) = \sum m_i$. However, when the matrix J is rank deficient, i.e. $rank(J) < \sum m_i$, the system can only have solution in the least squares sense and the resulting joint velocity $\dot{q}$ may become very large due to the singularity of J_i. For more detailed discussion on handling this singularity issue, we refer to [25]. A practical approach is to replace the pseudo-inverse $\hat{J}_i^+$ in the last step of the above algorithm by a singularity robust inverse, e.g. the damped least squares inverse [17].

We tested the task prioritization strategy on simulated human reaching obtained from the Carnegie Mellon University (CMU) human motion data base [5]. For this simulation, we used all 8 upper body descriptors which were grouped into three priority levels. The highest priority group corresponds to the Cartesian position of the waist. The medium priority group corresponds to the Cartesian position of the right and left wrists. Finally, the low priority group corresponds to the position of the elbows, shoulders, and neck. The results, depicted in Figure 6, illustrate that the highest priority descriptor has the lowest error. The non-zero error in the highest descriptor is attributed to the error introduced by the damping factor used for singularity robustness. The damping term is effective in handling task singularities as well as algorithmic singularities, but introduces errors in task position. To minimize the task error, an adaptive damping factor may be used. The two other priority levels also exhibited the expected error performance.



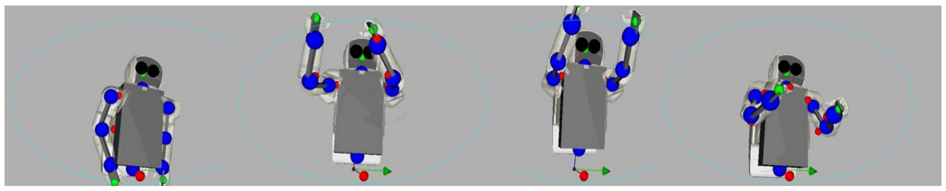


Fig. 6. Illustration of simulated tracking error for a reaching motion obtained from the CMU dataset. The task descriptors are assigned three priority levels. Lower figure illustrates snapshots of the reaching motion on the Asimo humanoid model.

4.4. *Joint limit avoidance constraints*

Joint limit avoidance is achieved by the proper selection of the weighting matrix W in Equation 10. Chan and Dubey [6] proposed joint limit avoidance based on a Weighted Least-Norm (WLN) solution. The WLN solution considers a candidate joint limit function, denoted by $H(q)$, that has higher values when joints near their limit and tends to infinity at the joint limits. One such candidate function is given by

$$H(q) = \frac{1}{4} \sum_{j=1}^{n} \frac{(q_{j,max} - q_{j,min})^2}{(q_{j,max} - q_j)(q_j - q_{j,min})},$$

where q_j represents the generalized coordinates of the j_{th} degree of freedom, and $q_{j,min}$ and $q_{j,max}$ are the lower and upper joint limits, respectively. The upper and lower joint limits represent the more conservative limits between the physical joint limits and the virtual joint limits used to avoid self collision as will be described in the Section 4.6. Note that $H(q)$ is normalized to account for variations in the range of motion. The gradient of H, denoted as ∇H, represents the joint limit gradient function, an $n \times 1$ vector whose entries point in the direction of the fastest rate of increase of H.

$$\nabla H = \frac{\partial H}{\partial q} = \left[\frac{\partial H}{\partial q_1}, \cdots, \frac{\partial H}{\partial q_n} \right]. \tag{13}$$

The element associated with joint j is given by

$$\frac{\partial H(q)}{\partial q_j} = \frac{(q_{j,max} - q_{j,min})^2 \ (2q_j - q_{j,max} - q_{j,min})}{4(q_{j,max} - q_j)^2 \ (q_j - q_{j,min})^2}.$$

The gradient $\frac{\partial H(q)}{\partial q_j}$ is equal to zero if the joint is at the middle of its range and goes to infinity at either limit. As described in [6], we define the joint limit gradient weighting matrix, denoted by W_{JL}, by an $n \times n$ diagonal matrix with diagonal elements w_{JL_j} $(j = 1 \cdots n)$. The matrix W in Equation 10 is constructed by W_{JL}, i.e. $W = W_{JL}$. The scalars w_{JLi} are defined by

$$\mathrm{w}_{JL_j} = \begin{cases} 1 + \left|\frac{\partial H}{\partial q_j}\right| & if \quad \Delta|\partial H/\partial q_j| \geq 0, \\ 1 & if \quad \Delta|\partial H/\partial q_j| < 0. \end{cases} \tag{14}$$

The term $\Delta|\partial H/\partial q_j|$ represents the change in the magnitude of the joint limit gradient function. A positive value indicates the joint is moving toward its limit while a negative value indicates the joint is moving away from its limit. When a joint moves toward its limit, the associated weighting factor, described by the first condition in Equation 14, becomes very large causing the motion to slow down. When the joint nearly reaches its limit, the weighting factor is near infinity and the corresponding joint virtually stops. If the joint is moving away from the limit, there is no need to restrict or penalize the motions. In this scenario, the second condition in Equation (14) allows the joint to move freely.

Figure 7 illustrates simulated results of the joint limit avoidance for a drinking motion sequence obtained from the CMU motion database. The solid spheres depicted on the ASIMO model corresponds to the actual task descriptor positions while the striped spheres represent the desired (or target) position of the task descriptors. The upper and lower joint limits are illustrated by the dashed lines, at 0 degrees and -177 degrees, respectively. When joint limit avoidance is turned off (left figure), the computed task descriptor (solid sphere) attached to the wrist can track the desired task descriptor (striped sphere) very well. However, the joint limit at the elbow is violated. When joint limit avoidance is turned on, the elbow joint limit is not violated. Since the elbow joint cannot flex beyond its joint limit, the computed wrist task descriptor does not track the desired wrist task descriptor.

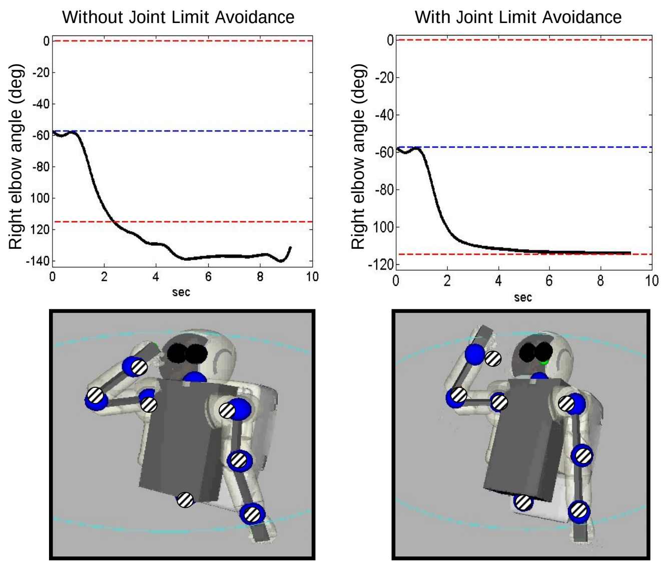


Fig. 7. Illustration of right elbow joint limit avoidance for a drinking motion from the CMU data-set. The upper and lower joint limits are illustrated by the dashed lines, at 0 degrees and -177 degrees, respectively.

4.5. *Joint Velocity Limits*

Joint velocity constraints are frequently handled by clamping the velocities when they reach their limit. While such an approach preserves the time required to execute the entire motion, it may not preserve the original trajectory profile. For fast human motions, the re-targeted robot motion profile may become significantly altered as a result of velocity clamping. We propose an alternative method to limit joint velocities by adaptively modulating the time between two successive time samples such that the motion profile q is re-scaled in time, but is not altered in its profile.

To simplify notation, we drop the subscript j, previously referred to quantities associated with joint j (i.e. $q = q_j$ and $\dot{q} = \dot{q}_j$). Let $\dot{q}_s \quad (s = 1 \cdots N)$ represent a length N sequence corresponding to a discrete time representation of $\dot{q}(t)$. In the discrete implementation of the algorithm, the discrete time sequence at sample $s+1$ is given by

$$t_{s+1} = t_s + \Delta t_s, \tag{15}$$

where Δt_s is time between sample s and $s+1$. To avoid velocity limits, we can replace the time sequence in Equation (15) with the following,

$$t'_{s+1} = \begin{cases} t'_s + \Delta t_s \ \epsilon_s & if \quad \epsilon_s \geq 1, \\ t'_s + \Delta t_s & if \quad \epsilon_s < 1. \end{cases} \tag{16}$$

where ϵ_s is a time-modulation factor defined by $\epsilon_s = \frac{|\dot{q}_s|}{\dot{q}_{lim}}$, and $\dot{q}_{lim}$ is the joint velocity limit associated with a particular degree of freedom. By definition, $\epsilon_s \geq 1$ implies that the joint velocities are equal or above their limits and corrective action is required by modulating (expanding) time between sample s and $s+1$. The resulting joint motion with the expanded timescale will conform to the velocity limits, without modifying the shape of the joint motion profile. Furthermore, $\epsilon_s < 1$ implies the joint velocities are below their limits and no time modulation is required. Equation 16 must be performed for each joint at each time-step. Note that each joint in a multi-degree of freedom robot may have a different modulation factor at each instant in time. In order to synchronize the sample time for all joints, a naive, yet simple solution would be to compute ϵ_s for all the joints, and select the largest among them for use in Equation 16.

The time modulation scheme presented above preserves the original motion profile, but may expand the total time required to execute the motion. In order to preserve the motion profile as well as the total execution time, it is possible to design alternative time modulation schemes where Δt_s is expanded when joint velocities exceed their limits and compressed when the joint velocities are below their limits. This may be performed in such a way that the overall execution time remains unchanged. Finally, it should be mentioned that the specification for time modulation may require that the first and second order time derivatives of q meet more strict smoothness and continuity requirements. In such a case, it is possible to use blending functions into the design of ϵ_s to maintain smoothness.

4.6. *Avoiding self collision*

Self collision avoidance may be categorized as one of two types: 1) collision between two connected segments, and 2) collision between two unconnected segment pairs. By connected segment pairs, we imply that the two segments are connected at a common joint and assume that the joint is rotational.

4.6.1. *Collision avoidance between two connected bodies*

If two segments are connected at a common rotational joint, i.e. connected segments, self collision may be handled by limiting the joint range as described in Section 4.4. Joint limits for self collision avoidance need not correspond to the physical joint limits; rather, they may be more conservative virtual joint limits whose values are obtained by manually verifying the bounds at which collision does not occur. Therefore, for two segments connected by a rotation joint, joint limit avoidance and self collision avoidance may be performed by using the same formulation presented in Section 4.4.

4.6.2. *Collision avoidance between unconnected bodies*

Consider two unconnected rigid bodies, i.e. bodies which do not share a joint, as shown in Figure 8. In general, Body A and body B may both be in motion. However, for simplicity of presentation and without loss of generality, suppose body A is moving toward a stationary body B. Let p_a and p_b represent the coordinates of the shortest distance $d (d \geq 0)$ between the two bodies, described in the base reference frame. Hereafter, we refer to p_a and p_b as collision points. The coordinates p_a and p_b can be obtained using a standard collision detection software. In this work, we use the SWIFT++ library [30].

Let $\hat{n}_a = \frac{p_b - p_a}{|p_b - p_a|}$ be the unit normal vector and $\vec{d} = d\ \hat{n}_a$ the vector from p_a to p_b. Consider a 3D virtual surface surrounding body A, shown by a dashed line in Figure 8. For every point on body A, its associated virtual surface point is located by the vector $\vec{d_c} = d_c\ \hat{n}$, where d_c is the critical distance, and $\hat{n}$ is the unit normal vector at the surface point. Let p_{vs_a} be the coordinates of a point on the virtual surface of A defined by

$$p_{vs_a} = p_a + d_c \hat{n}_a. \tag{17}$$

We define the region between the actual surface of body A and its virtual surface as the critical zone. If body B is stationary, we can redirect the motion at p_a to prevent collision in the critical zone. For now, we consider that the redirection is invoked when $d < d_c$. Later in this section, we will use a blending approach to adjust the initiation of the redirection. In our CLIK control framework, one way to

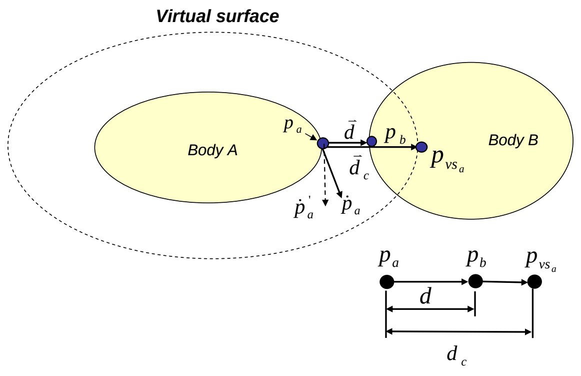


Fig. 8. Body A moving towards a fixed body B

control (or redirect) the motion of p_a is by modifying the trajectory of the desired task descriptor p_r. Let us specify a redirected motion of p_a by p'_a and its associated velocity by $\dot{p}'_a$. The question is, how shall we specify the magnitude and direction of $\dot{p}'_a$ to redirect the collision point to prevent the two bodies from penetrating deeper into the critical zone. There is no unique solution. The most straightforward, and perhaps conservative solution is to redirect the collision point in a direction opposite to the unit normal vector $\hat{n}_a$. A more effective strategy is to redirect the collision point so that it slides along a direction which is tangent to the surface at the collision point, as shown in Figure 8.

$$\dot{p}'_a = \dot{p}_a - < \dot{p}_a, \hat{n}_a > \hat{n}_a. \tag{18}$$

In theory, the above redirection vector will guide the collision point motion along the virtual surface boundary, producing a more natural motion toward the target.

To find the mapping between $\dot{p}'_a$ and $\dot{p}_r$, consider first the computation of the equivalent joint velocities which will redirect the collision point velocities along $\dot{p}'_a$. We compute the redirected joint velocity vector using the following mapping,

$$\dot{q}' = J_a^* \ \dot{p}'_a + SJ^*(\dot{p}_r + K \ e), \tag{19}$$

where $J_a = \partial p_a / \partial q$ is the Jacobian at the collision point and J_a^* is its weighted Damped Least Squares inverse. The matrix $S = diag(s_1 \cdots s_n)$ is a diagonal selection matrix where $s_i = 1$ when the i_{th} column of J_a has all zero entries and $s_i = 0$ elsewhere. The term $J^*(\dot{p}_r + K \ e)$ is simply the joint velocity solution obtained from Equation 9. The physical interpretation of Equation 19 is as follows. The first term determines the joint velocities needed to redirect the collision point velocities along $\dot{p}'_a$. Any zero column of J_a (all zero entries) implies that the associated degree of freedom does not contribute to the motion of the collision point. The second term in Equation 19 is the orthogonal complement of the first term which computes the entries for those joint velocities which do not affect the motion of the collision point(s). Intuitively, it would seem more appropriate to formulate Equation 19 using a two priority inverse kinematics strategy similar to the control of redundant manipulators [16]. In such a strategy, the first priority term corresponds to satisfying self collision avoidance by redirection (as in first term in Equation 19). Utilizing redundancy, the second priority term can be constructed to satisfy the requirements for tracking the task descriptors. In practice, this approach leads to jerky behaviors due to numerical instability to arrive at a prioritized solution when multiple colliding pairs enter and exit the critical zone. When there are multiple collision pairs, there is insufficient degrees of freedom to perform the secondary tasks. Numerical instability can also arise since collision points may be discontinuous.

Based on the collision free joint velocity commands computed from Equation 19, a redesigned position task descriptor trajectory may be computed as follows

$$\dot{p}'_r = J \ \dot{q}'. \tag{20}$$

The closed loop inverse kinematics equation with the modified parameters is given by

$$\dot{q} = J^*(\dot{p}'_r + K' \ e'), \tag{21}$$

where $e' = p'_r - p'$ and K' is an adaptively changing diagonal feedback gain matrix whose values decrease as the distance d decreases. Note that p'_r at the current time t may be computed by a first order numerical integration.

The instantaneous redirection $\dot{p}_a \rightarrow \dot{p}'_a$, as described above, produces a discontinuous first derivative of p_a at the boundary $d = d_c$. The discontinuity at $\dot{p}_a$ results in a discontinuity in $\dot{p}_r$, as given by the solution in Equation 20. To preserve first order continuity, we may blend the solutions of $\dot{p}'_r$ before and after redirection occurs. A blended solution to Equation 20 is given by

$$\dot{p}'_r = (1 - b)\ \dot{p}_r + b\ J\ \dot{q}', \tag{22}$$

where b is a suitable blending function such as,

$$b(d) = \frac{e^{-\alpha(d/d_c - \delta)}}{1 + e^{-\alpha(d/d_c - \delta)}}, \tag{23}$$

where α and δ are scalar parameters used to modulate the blending rate and shift of the blending function, respectively. Figure 9 shows the plot of b in relation to the ratio d/d_c for $\alpha = 15$. The blending function is plotted for $\delta = .5$ and $\delta = 1.0$. The parameter δ may be used to shift the distance d where blending is initiated and terminated. In the case $\delta = .5$, when $d > d_c$ the function $b(d) \approx 0$, implies that the second term in Equation 22 is effectively zero so that there is no redirection of the original task descriptor velocity (i.e. $\dot{p}'_r = \dot{p}_r$). At the other extreme, when $d = 0$, the function $b(d) = 1$, implies that the first term in Equation 22 is zero and the reference trajectory is altered in order to redirect the collision points along the tangent surface. To be more conservative, we may chose $\delta = 1.0$ in the blending function. This way, blending initiates even before the collision points reach their critical distance. The case when body A is stationary and body B is in motion is

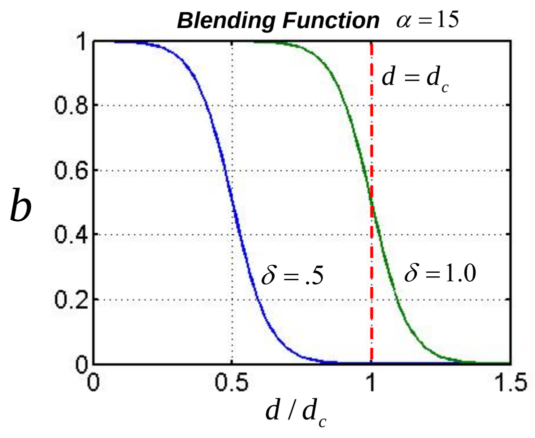


Fig. 9. Blending function at two values of δ

the dual of the problem considered above. When both body A and body B are in motion, we can specify the redirection vectors at the collision points p_a and p_b and use task augmentation to control both critical points.

Figure 10 illustrates snapshots of simulated retargeting results of a fast dancing motion with a full body twisting. The human data was obtained from the CMU motion capture database. These simulated results are generated using the humanoid robot ASIMO's model and geometry. As before, the eight upper body landmarks illustrated in Figure 2 were used as the detected key-points in the retargeting procedure. The top row illustrates the results without invoking the collision avoidance algorithm. The colliding body segments, detected using the SWIFT++ collision detection software, are highlighted by the diagonal striped pattern. The bottom row illustrates the results of the same motion when the collision avoidance was used.

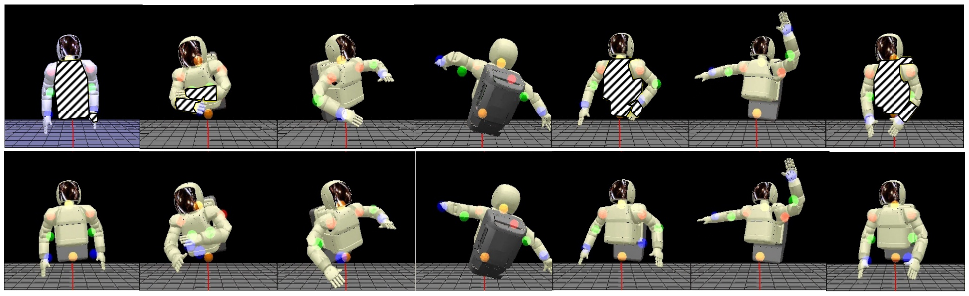


Fig. 10. Snapshots of simulated dancing motion with and without collision avoidance. Bodies which experience contact are represented by the diagonal striped pattern.

For the dancing sequence, Figures 11 and 12 show the minimum distance between collision points on the left hand and torso segment pairs, and left hand and right hand collision pairs, respectively. The minimum distances are plotted with and without using the collision avoidance algorithm. Without collision avoidance, the collision points attached to the left hand and torso segment penetrate the collision zone and eventually collide between frames 470 and 505 as shown in Figure 11. Note that a negative distance implies collision and penetration of the two bodies. Penetration distance is clamped when penetration of the two bodies is beyond $-2.5\ cm$. When collision avoidance is turned on, contact between the two segments does not occur. The blending parameter was set at $\delta = .5$ such that blending is initiated at the critical distance of $d_c = 5.0\ cm$; therefore, collision points are not fully redirected at the virtual surface. Redirection is gradual, and penetration into the critical zone occurs. However, the two bodies do not collide. Figure 12 shows more dramatic contact between the left hand segment and the torso segment. The collision avoidance algorithm can successfully avoid penetration.

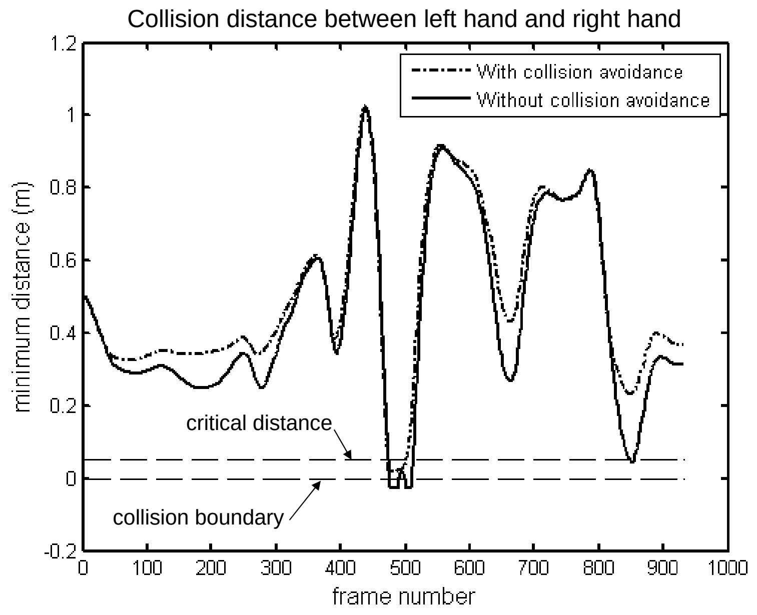


Fig. 11. Minimum distance between left and right hand collision points for a dancing motion. Critical zone is set at .05 meters, and depicted by the dashed line.

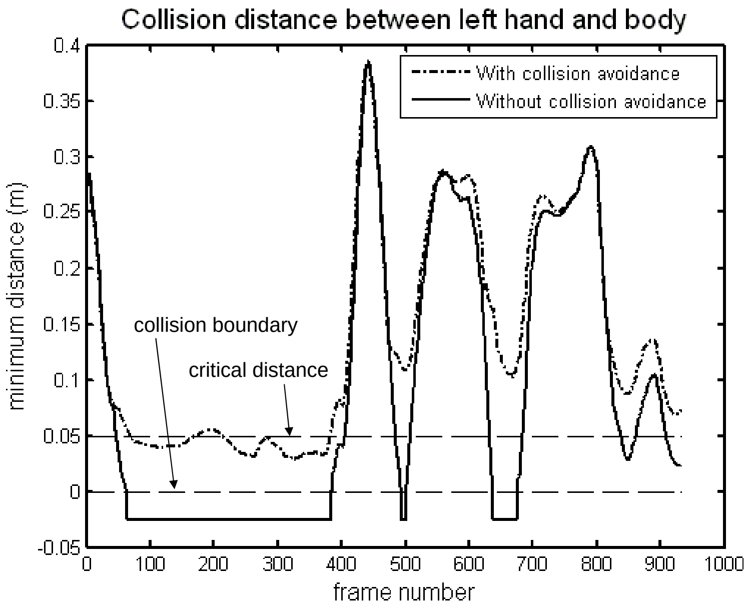


Fig. 12. Minimum distance between left hand and torso collision points for a dancing motion. Critical zone is set at .05 meters, and depicted by the dashed line.

4.7. *Balance Control*

The presented control scheme does not yet consider the constraints that are required to maintain balance during standing and walking. These aspects are not handled within the retargeting framework, but rather by a separate walking and balancing controller that is described in [11,12]. In detail, the retargeted motion is commanded

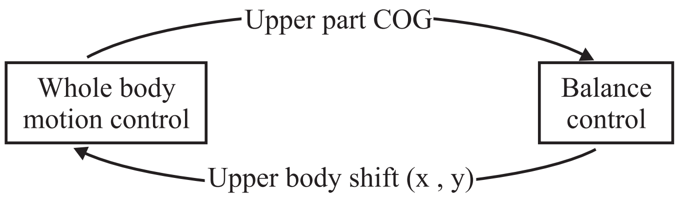


Fig. 13. Separation of balance- and whole body posture control.

to the whole body motion controller. The motion generated by the whole body controller will cause some momentum and moment of momentum from a desired reference. This deviation is compensated by the ZMP based balance controller by shifting the upper body in forward- and lateral direction. As depicted in Figure 13, the whole body control and the ZMP control operate cooperatively.

To account for the body shift, the upper body translational degrees of freedom are incorporated in the kinematic model of the robot. However, they are not actively driven, but rather considered as external input into the controller equations of the whole body control. Whole body control and ZMP control are coupled through momentum and state feedback, which turns out to be an efficient way to separate these controllers.

5. Motion Interface

We use a motion interface to provide a communication link and command interface between off-board computations to generate the retargeted joint commands and the on-board real time control. The motion interface provides a comprehensive way to give motion commands to the robot, without having the user to care about issues such as synchronization, delays, or on-board control for maintaining balance. As a safety measure, there is also an on-board self collision avoidance in the real time control loop with details given in [8,26]. Such an interface is desirable since the real-time implementation requires synchronization between critical control processes that may not be satisfied dependably with a network connection. Issues like balance control, on-board self collision avoidance, and other critical aspects are handled within the real-time controller. The motion interface has been successfully used in various other applications, as for instance in [4].

6. Experimental Results

Experiments were performed on the Honda humanoid robot, Asimo, using a single time-of-flight range image sensor [1], to obtain depth images at approximately 15 frames per second. As described previously, the eight upper-body key-points shown in Figure 2 were detected and tracked at approximately 10 frames per second. The data was further processed and the retargeting module then generates collision free joint commands for the Humanoid Robot Asimo at 100 Hz. A socket program sends the joint commands to the motion interface, described in Section 5, over a wireless network. The whole body motion interface then communicates these joint space coordinates to low level controllers on the robot, including the balance controller, that run on a dedicated real time processing computer onboard the robot, through UDP sockets. The Experimental setup of the entire pipeline is illustrated Figure 14.

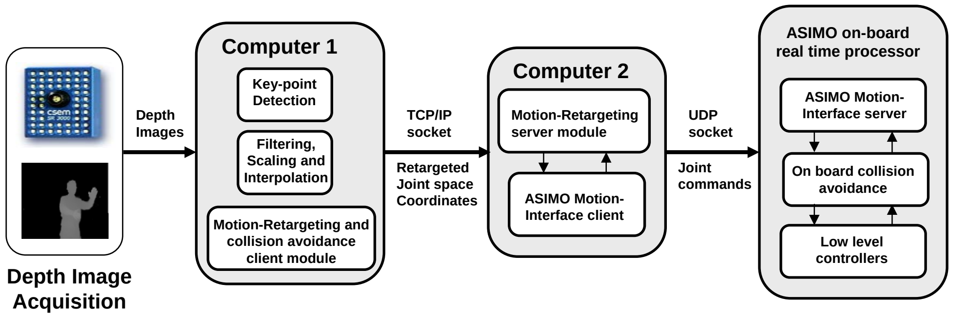

Depth Image Acquisition
Depth Images
Computer 1
Key-point Detection
Filtering, Scaling and Interpolation
Motion-Retargeting and collision avoidance client module
TCP/IP socket
Retargeted Joint space Coordinates
Computer 2
Motion-Retargeting server module
ASIMO Motion-Interface client
UDP socket
Joint commands
ASIMO on-board real time processor
ASIMO Motion-Interface server
On board collision avoidance
Low level controllers

Fig. 14. Set-up for experiments

Figure 15 illustrates experimental results of online retargeting from three different demonstrators performing various motions which contains in-plane and out-of-plane movements. The sequence in the first two rows depict snapshots from a Taiji dance motion. The last row illustrates simple exercise motions which demonstrates that the balance controller makes slight adjustments to the waist and the lower body to accommodate for the upper body movements. This is especially apparent in the first image of the last row. In all experiments, Asimo replicates the motion fairly well while enforcing all kinematic constraints. A delay of approximately .25 seconds is observed in the tracking of the human motion by Asimo during experiments, although visual processing and retargeting algorithms performed with no software latency. This delay is attributed to software overhead in the communication and control protocols. Efforts to resolve these latency issues are in progress.

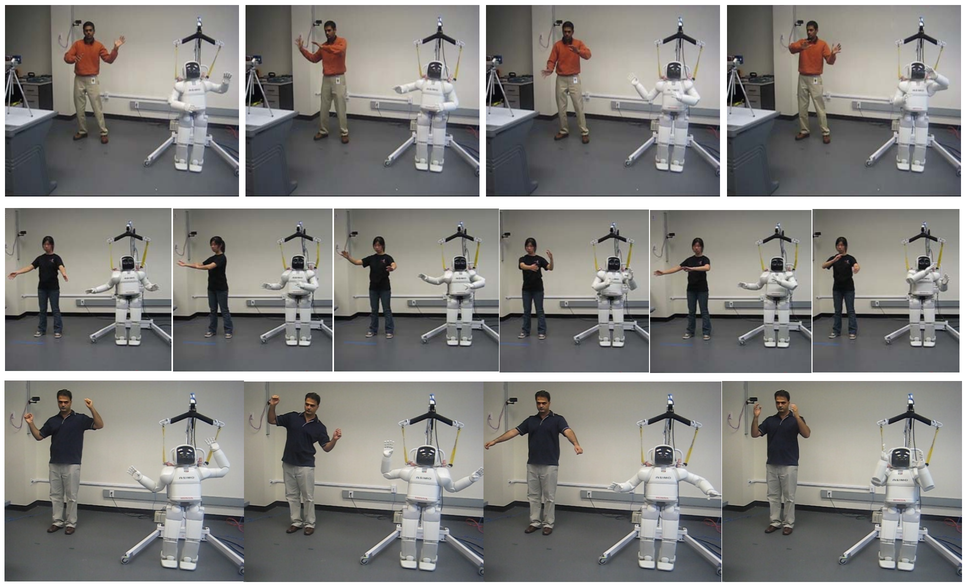


Fig. 15. Snapshots from online motion retargeting to Asimo

7. Summary and Future Work

Although several systems have previously demonstrated human to humanoid motion retargeting, the approach described in this paper is distinct in several ways. Notably, the three major contributions are as follows. First, the proposed retargeting framework relies on human motion descriptors (or task descriptors) obtained from a marker-less vision algorithm using a single time-of-flight camera. These task descriptors are noisy, sparse, and represent only a few key feature points on the human body. We reported retargeting results based on eight upper body task descriptors; however, the proposed formulation can handle an arbitrary number. The algorithm is suitable when there is redundant degrees of freedom as well as when the system is over-constrained. In fact, for many of the motions tested, we observed that utilizing as few as four task descriptors (waist, two hands, head) could reproduce realistic and natural looking robot motions. This attribute enables flexibility in sensing and instrumentation required to acquire human motion, as well as flexibility in controlling the robot by a limited number of task descriptors.

The second important contribution is the online self collision avoidance algorithm. This problem is particularly challenging in humanoid motion control since for a given motion, several segments can simultaneously collide. We presented a robust method which can cope with fast motions where multiple collisions can simultaneously occur. Unlike many existing collision avoidance algorithms such as those based on virtual forces or potential functions, the proposed method does not require parameter tuning and is not subject to numerical instabilities such as those observed based on null-space projections.

The third important contribution of this paper is the system and algorithmic integration to create a unified framework for online motion retargeting with kinematic constraints. Although certain individual components used in this paper are based on previously developed algorithms, the integration and modification of those algorithms to create a unified framework is novel. The complete pipeline has been tested and verified experimentally on the Asimo humanoid robot.

Two notable extensions of the current framework involve the inclusion of constraints to enforce torque limits and development of a more sophisticated balance controller. Currently, the upper body retargeting algorithm relies on Asimo's existing balance controller in order to make balance adjustments by modulating the pelvis position. No provisions are made for the robot to automatically take steps in order to sustain balance due to large disruption to the upper body. Stepping is an essential component of a complete whole body balance controller as has been demonstrated by Nakaoka et. al [18]. We are currently considering incorporating stepping strategies into our whole body motion retargeting framework.

References

1. *Swiss Ranger SR-3000 3D time of flight camera*. http://http://www.swissranger.ch/.
2. H. Bekkering, A. Wohlschlaeger, and M. Gattis. Imitation of gestures in children is goal directed. *Quarterly Journal of Experimental Psychology*, 53A(1):153–164, 2000.
3. N. Bernstein. *The coordination and regulation of movements.* Pergamon, London, 1967.
4. B. Bolder, M. Dunn, M. Gienger, H. Janssen, H. Sugiura, and C. Goerick. Visually guided whole body interaction. In *Proceedings of the Int. Conf. on Robotics and Automation*, Rome, Italy, 2007.
5. Carnegie Mellon University. CMU graphics lab motion capture database. Internet page. http://mocap.cs.cmu.edu/.
6. T. F. Chan and R. V. Dubey. A weighted least-norm solution based scheme for avoiding joint limits for redundant joint manipulators. *IEEE Transactions on Robotics and Automation*, 11(2), 1995.
7. K. J. Choi and H. S. Ko. On-line motion retargeting. *Journal of Visualization and Computer Animation*, 11(5):223–235, 2000.
8. M. Gienger, H. Janssen, and C. Goerick. Task-oriented whole body motion for humanoid robots. In *Proceedings of the 2005 5th IEEE-RAS International Conference on Humanoid Robots*, pages 238–244, Los Angeles, USA, 2005.
9. M. Gleicher. Retargeting motion to new characters. In *Proceedings of SIGGRAPH98*, pages 33–42, ACM, New York, 1998.
10. F. Guenter and A. G. Billard. Using reinforcement learning to adapt an imitation task. In *Conference on Intelligent Robots and Systems (IROS07)*, pages 1022–1027, San Diego, CA, 2007.
11. M. Hirose, Y. Haikawa, T. Takenaka, and K. Hirai. Development of humanoid robot ASIMO. In *IEEE/RSJ International Conference on Intelligent Robots and Systems - Workshop 2*, Hawaii, USA, 2001.
12. Honda Motor Corp. The Honda Humanoid Robot Asimo. Internet page. http://www.world.honda.com/ASIMO.
13. S. Kagami, F. Kanehiro, Y. Tamiya, M. Inaba, and H. Inoue. Autobalancer: An online dynamic balance compensation scheme for humanoid robots. In *Int. Workshop Alg. Found. Robot.(WAFR)*, Lausanne, Switzerland, 2000.

14. J. Lee and S. Y. Shin. A hierarchical approach to interactive motion editing for human-like figures. In *Proceedings of SIGGRAPH99*, pages 39–48, ACM, New York, 1999.
15. J. Luh, M. Walker, and R. Paul. Resolved-acceleration control of mechanical manipulators. *IEEE Transactions on Automatic Control*, 25:468–474, 1980.
16. Y. Nakamura. *Advanced Robotics, Redundancy and Optimization.* Adisson-Wesley, 1991.
17. Y. Nakamura and H. Hanafusa. Inverse kinematic solution with singularity robustness for robot manipulator control. *ASME J. Dyn. Sys. Meas., Contr.*, 108(3):163–171, 1986.
18. S. Nakaoka, A. Nakazawa, F. Kanehiro, K. Kaneko, M. Morisawa, H. Hirukawa, and K. Ikeuchi. Learning from observation paradigm: Leg task models for enabling a biped humanoid robot to imitate human dances. *Int. J. Robotics Research*, pages 829–844, 2007.
19. A. Nakazawa, S. Nakaoka, K. Ikeuchi, and K. Yokoi. Imitating human dance motions through motion structure analysis. In *Intl. Conference on Intelligent Robots and Systems (IROS)*, pages 2539–2544, Lausanne, Switzerland, 2002.
20. N. Pollard, J. K. Hodgins, M. Riley, and C. Atkeson. Adapting human motion for the control of a humanoid robot. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA '02)*, May 2002.
21. B. Robins, K. Dautenhahn, R. Boekhorst, and A. Billard. Robotic assistants in therapy and education of children with autism: can a small humanoid robot encourage social interaction skills? *Universal Access in the Information Society (UAIS)*, 4(2):105–120, 2005.
22. A. Safonova, N. Pollard, and J. Hodgins. Optimizing human motion for the control of a humanoid robot. In *Int. Symp. on Adaptive Motion of Animals and Machines (AMAAM2003)*, Kyoto, Japan, 2003.
23. S. Schaal. Learning from demonstration. In M. Mozer, M. Jordan, and T. Petsche, editors, *Advances in Neural Information Processing Systems*, chapter 9, pages 1040–1046. MIT Press, 1997.
24. L. Sentis and O. Khatib. A whole-body control framework for humanoids operating in human environments. In *Proc. of Int. Conf. on Robotics and Automation (ICRA06)*, Orlando, FL, 2006.
25. B. Siciliano and J. Slotine. A general framework for managing multiple tasks in highly redundantrobotic systems. In *International conference on Advanced Robotics*, volume 2, pages 1211–1216, Pisa, Italy, 1991.
26. H. Sugiura, M. Gienger, H. Janssen, and C. Goerick. Real-time collision avoidance with whole body motion control for humanoid robots. In *IEEE Int. Conf. on Intelligent Robots and Systems (IROS 2007)*, 2007.
27. S. Tak and H. Ko. A physically-based motion retargeting filter. *ACM Trans. on Graphics*, 24(1):98–117, 2005.
28. S. Tak, O. Song, and H. Ko. Motion balance filtering. *Comput. Graph. Forum. (Eurograhics 2000)*, 19(3):437–446, 2000.
29. A. Ude, C. Atkeson, and M. Riley. Programming full-body movements for humanoid robots by observation. *Robotics and Autonomous Systems*, 47:93–108, 2004.
30. UNC Chapell Hill: Swift++ Library. Speedy walking via improved feature testing for non-convex objects. Internet page. http://www.cs.unc.edu/ geom/SWIFT++/.
31. Y. Zhu, B. Dariush, and K. Fujimura. Controlled human pose estimation from depth image streams. In *CVPR Workshop on Time of Flight Computer Vision*, Anchorage, Alaska, 2008.