A Generative Human-Robot Motion Retargeting Approach using a Single Depth Sensor

Sen Wang[1,2] Xinxin Zuo[1,2] Runxiao Wang[1] Fuhua Cheng[2] Ruigang Yang[2]

***Abstract*— The goal of human-robot motion retargeting is to let a robot follow the movements performed by a human subject. This is traditionally achieved by applying the estimated poses from a human pose tracking system to a robot via explicit joint mapping strategies. In this paper, we present a novel approach that combine the human pose estimation and the motion retarget procedure in a unified generative framework.**

A 3D parametric human-robot model is proposed that has the specific joint and stability configurations as a robot while its shape resembles a human subject. Using a single depth camera to monitor human pose, we use its raw depth map as input and drive the human-robot model to fit the input 3D point cloud. The calculated joint angles of the fitted model can be applied onto the robots for retargeting. The robot's joint angles, instead of fitted individually, are fitted globally so that the transformed surface shape is as consistent as possible to the input point cloud. The robot configurations including its skeleton proportion, joint limitation, and DoF are enforced implicitly in the formulation. No explicit and pre-defined joints mapping strategies are needed.

This framework is tested with both simulations and real robots that have different skeleton proportion and DoFs compared with human to show its effectiveness for motion retargeting.

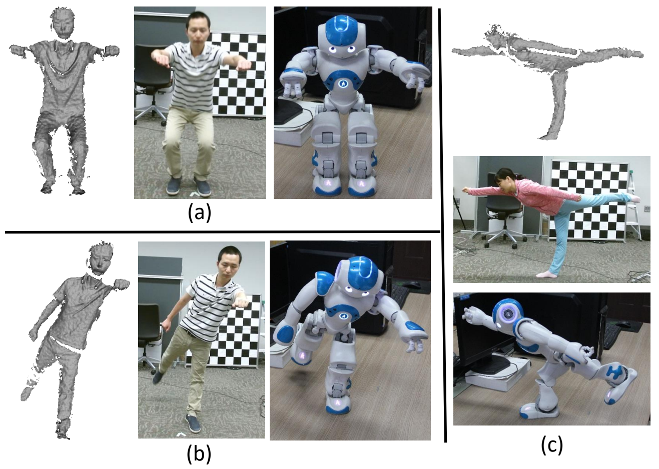


Fig. 1: Motion retargeting results of three different poses. The mesh and color image are captured by Kinect V2. These poses retargted to NAO robot using our proposed approach.

I. Introduction

Nowadays social robots have become more intelligent along with the progress in object recognition and speech recognition. However the limited motion generating strategy is one bottleneck that prevents them from more widespread use. It is still an active research topic for which many approaches have been proposed. One effective way to generate motion for robots is to let a robot mimic the movement of a human, i.e., human-robot motion retargeting. The goal is to drive a humanoid robot to move in a natural way provided with the joint movements or positions captured from human. This is a hot topic in both robotics and other areas, as motion retargeting can be widely used in robot simulation [1] and also in virtual characters animation [2]. In this paper, we pay our attention on how to make robots imitate the human motion.

Previously, the joints position obtained from a motion capture (Mocap) system [3] or Kinect skeleton tracking algorithm [4] is typically considered as the input of a motion retargeting procedure. After that the joint movements or the end-effector positions are applied onto robots via some pre-defined mapping strategies between human and robots. One straightforward way for motion retargeting is to apply inverse kinematics (IK) to end-effector positions given the tracked human joint positions. Another kind of solution is to preserve the joint angles, which places a major role for defining mapping criteria.

These kinds of methods are not always sufficient since the skeleton of a robot usually differs from the skeleton of a human, such as the proportion of their limbs. Therefore we need to define the mapping between the target and the source model and this mapping can not be reused in robot models with different configurations. Another drawback for these methods is that we cannot enforce joint angles and end positions constraints simultaneously. This will cause the dissimilarity of robot movements and human movement. Moreover, they have quite different joint limitations and robots usually have less degree of freedoms (DoFs) than human beings. Stability is another essential problem that needs to be dealt with while imitating human motion. These requirements pose more challenges for human-robot motion retargeting.

In this paper, rather than dealing with motion tracking and retargeting separately, we present a novel method that combines these two procedures in a united framework. We use the captured 3D point cloud from a single consumer depth sensor such as Kinect as our input instead of the computed skeleton, and the human subject does not wear any markers.

For the robot model representation, it is usually modeled

[1]Northwestern Polytechnical University, Xi'an 710072, P.R.China {wangsen1312, xinxinzuo2353}@gmail.com, wangrx@nwpu.edu.cn

[2]Univeristy of Kentucky, Lexington, KY 40506, USA {cheng,ryang}@cs.uky.edu

as joint or skeleton; in contrast we choose to use a parametric human-robot template to represent the joints together with 3D surface shape. More specifically, we propose a *HUMROB* model that has the joint configurations of the target robot and the 3D shape same as the source human subject. The robot is modeled as 3D mesh with bones and joints embedded inside the surface. Our goal is to compute the joint angles of the robot by driving the template model so that the surface shape fits as much as possible the point cloud acquired from depth sensor. In this way, we can get the joint angles for the robot that maintain the pose similarity. Besides, the stability and joint limitations are enforced in the objective function naturally. Since we do not need an explicit joint or position mapping strategies, our method can be applied to robots with different configurations quite conveniently. We classify our method as a generative motion retargeting approach as compared with previous methods that focus on developing various skeleton mapping strategies.

As far as we know, our technique is the first one that uses a generative unified framework to achieve motion retargeting with one single depth sensor. Our algorithm is validated with both simulation and real robots with different skeleton proportions from human. Satisfactory and stable motions have been generated even under some extreme poses. Some examples are shown in Fig. 1.

II. Related Work

Motion retargeting is a hot topic in robotics that has been used in gaming and motion generation. For previous motion retargeting methods, two separate procedures are included: motion capture of source and then motion retargeting for target. For a detailed review of various algorithms, especially for human-robot motion retargeting, we refer the reader to the survey [5]. Some other methods have demonstrated high accuracy performance while considering kinematic and dynamic constraints such as joint limits, self collisions, torque limits, balance and so on [6], [7], [8]. In this section, we review only the most related work focusing on human-robot motion retargeting including pose estimation and motion retargeting.

A. Human Pose Estimation

Human pose estimation is mainly categorized into marker based method, inertial based system, and markerless method regarding to the device that has been used.

Marker based system. Marker based systems require the user to wear a special suit with markers attached to fixed body position, such as Vicon™ systems [3] and Optitrack™ system.

Inertial based system. Xsens MVN™ motion capture system [9], which consisting of inertial sensors attached to the individual body segments to get the precise position.

Although these two kinds of systems can offer high accuracy and a rate of frames, precise marker position and huge cost is an obstacle for widespread use.

Markerless system. Motion capture with only color cameras is still an unsolved problem for computer vision community considering the ill-posed condition [10]. With the availability of consumer depth sensors such as Kinect, it is easy to get the human skeleton using its SDK [11], [12] which shows more accuracy than color only methods. Apart from the default use with original SDK, researchers focus on how to get the human pose more precise. The papers mainly include discriminative or generative approach for model based pose estimation. Existing discriminative approaches either performed body part detection by identifying salient points of the human body [13], or relied on classifiers or regression machines trained offline to infer the joint locations [14]. The generative approaches fit a template to the observed data. Ye et al. [15] relates the observed data with model template using Gaussian Mixture Model, without explicitly building point correspondences.

B. Motion Retargeting

The challenge for motion retargeting can be summarized into two aspects: 1) Source and target has similar or same topology but with different geometry, that means the source/target is expressed as similar or even the same hierarchy link of joints, but the ratio length of bones is different from each other. 2) Source and target has different topology, that is a more complex situation and also different geometry.

Different geometry. For the different geometry but same topology, it is common to employ IK solvers integrated with soft constraints enforced in an object function. Also, some hard constraints are employed to define specific rules for the motion that must be maintained [4], [9], [16].

Different topology. Source and target are identified as topologically different if they do not have the similar skeleton structure, some methods have been proposed, among which exampled-based methods [17], [18] are commonly used. This method based on a number of motion mappings, several key poses and correspondences are pre-defined.

Unlike the traditional separated pose estimation and mapping procedure, in this paper we propose a unified framework combines the two procedures using a single depth sensor. We can deal with motion retargeting both the different geometry and different topology case without special treatment.

III. Background

We first introduce some background knowledge that is essential for our approach.

A. Gaussian Mixture Model

We give a brief introduction to GMM and its solution using EM algorithm, which is the fundamental techniques for our framework. More details can be found in [19].

Assume that we have N source points, $S_{N*D} = (\boldsymbol{s}_1, \ldots, \boldsymbol{s}_N)^T$ and M target points, $T_{M*D} = (\boldsymbol{t}_1, \ldots, \boldsymbol{t}_M)^T$. D is the dimensionality of the point set, so $D = \mathbf{3}$ in this paper.

We assume that the source points follows the GMM whose centroids are depend on the target points. The probability of each source point is expressed as,

$$p(\boldsymbol{s}) = (1-\mu)\sum_{m=1}^{M}\frac{1}{M}p(\boldsymbol{s}|\boldsymbol{t}_m) + \mu\frac{1}{N} \tag{1}$$

where

$$p(\boldsymbol{s}|\boldsymbol{t}_m) = \frac{1}{(2\pi\sigma^2)^{\frac{3}{2}}}\exp(-\frac{||\boldsymbol{s}-\boldsymbol{t}_m||^2}{2\sigma^2}) \tag{2}$$

The outliers are considered as subjecting to the uniform distribution with μ as the approximation for the percentage of outliers.

Next we reparametrize the GMM centroid locations by a set of parameters ϕ and estimate them by minimizing the negative log-likelihood function,

$$\begin{aligned} E(\phi,\sigma^2) &= -\sum_{n=1}^{N}\log\sum_{m=1}^{M+1}P(\boldsymbol{t}_m)p(\boldsymbol{s}_n|\boldsymbol{t}_m(\phi)) \\ &= -\sum_{n=1}^{N}\log(\sum_{m=1}^{M}\frac{1-\mu}{M}p(\boldsymbol{s}_n|\boldsymbol{t}_m(\phi)) + \frac{\mu}{N}) \end{aligned} \tag{3}$$

The energy function is usually solved using the EM algorithm in an iterative manner as described in [20]. We summarize the E-step and M-step briefly here.

During the **E-step**, given the values of parameters estimated from M-step of the previous iteration, we can then use the Bayes' rules to compute a posteriori probability distribution for the target data points given the source data,

$$p^{old}(\boldsymbol{t}_m(\phi)|\boldsymbol{s}_n) = \frac{\exp(-\frac{||\boldsymbol{s}_n-\boldsymbol{t}_m^{old}(\phi)||^2}{2(\sigma^{old})^2})}{\sum_{i=1}^{M}\exp(-\frac{||\boldsymbol{s}_n-\boldsymbol{t}_i^{old}(\phi)||^2}{2(\sigma^{old})^2})+c}, \tag{4}$$

where $c = \left(2\pi(\sigma^{old})^2\right)^{\frac{3}{2}}\frac{1-\mu}{\mu}\frac{M}{N}$.

To simplify the expression, we let p^{old} to stand for $p^{old}(\boldsymbol{t}_m(\phi)|\boldsymbol{s}_n)$ in the following sections.

During the **M-step**, the parameters are updated via minimizing the following complete negative log-likelihood function.

$$Q(\phi,\sigma^2) = \frac{1}{2\sigma^2}\sum_{n,m=1}^{N,M}p^{old}||\boldsymbol{s}_n-\boldsymbol{t}_m(\phi)||^2 + \frac{\mathbf{3}}{2}N_{\boldsymbol{P}}\log\sigma^2, \tag{5}$$

Where $N_{\boldsymbol{P}} = \sum_{n,m=1}^{N,M}p^{old}$. The value of all the parameters will converge after several iterations.

B. Joint and Vertex transformation

To better represent the posture of whole body, we use the classical twists exponential map for joint transformation and adopt skeleton-subspace deformation for vertex transformation.

1) Joint transformation: For articulated model such as human and robot, pose can be represented by a set of twists $\hat{\xi}\phi \in SO(3)$, which is the exponential mapping that can be generalized to the Euclidean group ($SE(3)$) [21]. We give more details in the following.

In general, an articulated body transformation is represented as the exponential of the twist $T \in SE(3)$ as follows,

$$T = \exp(\hat{\xi}\phi) \tag{6}$$

In homogeneous coordinates, the twist $\hat{\xi} \in se(3)$ can be expressed as follows,

$$\hat{\xi} = \begin{bmatrix}\hat{\omega} & v\\ 0 & 0\end{bmatrix} \in \mathbb{R}^{4\times4}, \tag{7}$$

where $\hat{\omega} \in so(3)$ is 3×3 skewed-symmetric matrix converted from a 3D vector $\omega \in \mathbb{R}^3$, which is the direction of the rotation axis. The 3D vector v determines the location of the rotation axis and the amount of translation.

Assume that robot has P joints, including the global transformation and rotation, the whole body motion namely the pose is controlled by these parameters $\Phi := (\phi_0, \phi_1 \ldots, \phi_P)$. With the twists exponential map, final transformation for each joint k is represented as,

$$T_k = \prod_{p=0}^{P}\exp(\kappa_{pk}\hat{\xi}_p\phi_p), \tag{8}$$

where $\hat{\xi}_0\phi_0$ is the global transformation and rotation for the root that consists of the six parameters. $\kappa_{pk} = 1$ if p is the hierarchical parent of the joint k, otherwise $\kappa_{pk} = 0$.

2) Vertex transformation: When we get the joint transformation under a certain pose Φ, the position of each vertex v of template mesh is computed by skeleton-subspace deformation (also known as "LBS") method [22],

$$v_i(\Phi) = \sum_{p=1}^{P}\omega_{ip}T_p(\Phi)v_p^0, \tag{9}$$

where ω_{ip} is the skinning parameters attaching the ith vertex to the underlying the joint p, and v_i^0 stands for the position for each vertex in its reset pose.

IV. Approach

Our goal is to retarget the motion captured from human subjects to robots in a generative way without any predefined joint mapping. The overall pipeline is shown in Fig. 2. There are mainly two steps. First, we present our

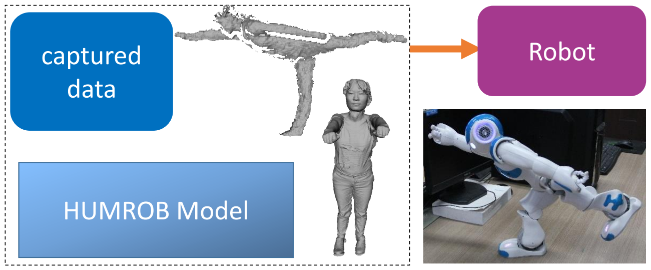


Fig. 2: System Pipeline. We use captured depth data as input. First, we build up the HUMROB model, which is then deformed to fit the captured data and the joint angles are computed. Finally, the pose is applied onto the robot.

method to create a parametric *HUMROB* model in section IV-A; then the template model is deformed to fit the captured depth map of the source human subject as described in section IV-B. We enforced joint limitations and stability constraints during the deformation. Finally, the joint angles computed via the deformation can be applied directly onto the robots for motion retargeting.

A. *HUMROB Model*

Considering our goal of fitting robot model to captured point cloud from depth sensor, first we need to build a personalized 3D Human-Robot model, which is a parametric model that can be deformed according to joint angles and positions. Also, the robot configurations, which are usually different from humans, need to be embedded into its parametric model. Several steps are performed to get this personalized parametric *HUMROB* model as shown in Fig. 3.

The first step is to build a personalized 3D model for human source subjects. Nowadays, there are several successful approaches that can be used to get the 3D model. For example, we can use multiple depth sensors and then fuse the depth maps captured from multiple view directions [23]. Besides, a single depth sensor is also sufficient to provide the complete 3D human model [24].

Then given the 3D model of the human subject, the next step is to parametrize the model using the generic parametric human model. In more details, we use the personalized 3D model as a target, and the limb length and the geometry of the generic template is adapted to fit the target model, serval methods can be applied, such as [25]. As with the adaptation, the configurations of generic parametric model are adjusted to make the model align with the 3D personalized model of the human subject. After the alignment, the skinning weight transfer procedure is followed where k-nearest neighbor(k-NN) search is implemented between generic template and personalized model to obtain the raw skinning weight. Finally, we use the laplacian smooth to smooth the transferred skinning weight. By the end of this step we get the personalized parametric model automatically.

Finally, we apply the robot configurations including the limb proportion and its joints limitation to the personalized parametric model we just have obtained. In this step, the joint positions for each limb will get adjusted and we will also change the DoFs of particular joint according to the robot we intend to retarget to.

B. *Motion Retargeting*

Now we have got the *HUMROB* model, then we can perform motion retargeting under the motion tracking framework by fitting this parametric model to captured depth data. The framework consists of two fundamental techniques namely the Gaussian Mixture Model and the joint/vertex transformation described in Sec. III-A and Sec. III-B.

Another thing we need to point out is that we choose to do the tracking and motion retargeting continuously along the sequence rather than selecting key frames and performing retargeting frame by frame as often used in previous works.

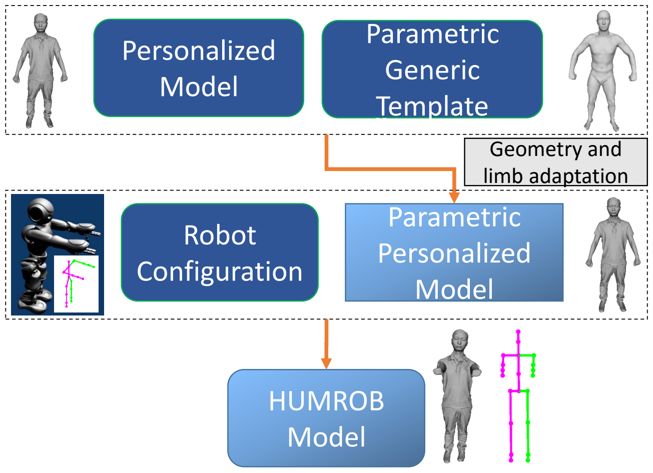

Personalized Model
Parametric Generic Template
Geometry and limb adaptation
Robot Configuration
Parametric Personalized Model
HUMROB Model

Fig. 3: Pipeline for building up personalized parmateric HUMROB model

First, suppose that the robot has J joints and we have got the pose (the joint angles in our case) at time t denoted as $\Theta^t := (\theta_0^t, \theta_1^t \ldots, \theta_J^t)$, we intend to compute the changes for the pose $\Delta\Theta$ from time t to $t+1$ as follows,

$$\Theta^{t+1} = \Theta^t + \Delta\Theta \tag{10}$$

Under twist-based parametrization, the transformation for any joint k at time $t+1$ can be expressed as,

$$T_k^{t+1} = \prod_{j=0}^{J} \exp(\kappa_{jk}\hat{\xi}_k(\theta_j^t + \Delta\theta_j)) \tag{11}$$

1) Energy formulation: To compute the pose expressed by Θ, we formulate the energy function as below,

$$E = E(\Theta, \sigma^2) + \lambda_\perp E_\perp(\Theta, \sigma^2) + \lambda_r E_r(\Delta\Theta) \tag{12}$$

The **first term** is fitting cost that measures the distance of deformed *HUMROB* template model from captured point cloud, which is minimized to enforce pose similarity. To give more mathematical details, the observed points from the capture human pose are denoted as ($S_{N*\mathbf{3}} = (\boldsymbol{s}_1, \ldots, \boldsymbol{s}_N)^T$), and the template points sampled from the *HUMROB* model are represented as a vertex set ($V_{M*\mathbf{3}} = (\boldsymbol{v}_1, \ldots, \boldsymbol{v}_M)^T$). If we take $V_{M*\mathbf{3}}$ as the centroids of GMM model, then $S_{N*\mathbf{3}}$ can be seen as the observed data that should be generated from the centroids via GMM model as described in Sec.III-A.

Therefore the vertex position fitting cost can be expressed as,

$$E(\Theta, \sigma^2) = \frac{1}{2\sigma^2} \sum_{n,m=1}^{N,M} p_{mn} ||\boldsymbol{s}_n - \boldsymbol{v}_m(\Theta)||^2 + \frac{\mathbf{3}}{2} N_{\boldsymbol{P}} \log\sigma^2, \tag{13}$$

Similar to p^{old} in Eq. 4, the p_{mn} can be seen as the probability of $\boldsymbol{s}_n$ relating to $\boldsymbol{v}_m$, we give more details in section V-B. $v_m(\Theta)$ is calculated by substituting the $T_j(\Theta)$ in Eq. 9 with Eq. 11, and then we get the position for each vertex computed as,

$$v_m(\Theta) = \sum_{j=1}^{J} \omega_{mj} \Big(\prod_{j=0}^{J} \exp(\kappa_{jk}\hat{\xi}_k(\theta_j^t + \Delta\theta_j)) \Big) v_m^0, \quad (14)$$

The **second term** in Eq. 12 is another fitting term that ensures the normal consistency between deformed template model and the observed depth data. The cost function for this term is given as,

$$E_{\perp}(\Theta, \sigma^2) = \frac{1}{2\sigma^2} \sum_{n,m=1}^{N,M} p_{mn} ||\boldsymbol{s}_n^{\perp} - \boldsymbol{v}_m^{\perp}(\Theta)||^2 + \frac{\mathbf{3}}{2} N_{\boldsymbol{P}} \log\sigma^2, \quad (15)$$

where $\boldsymbol{s}_n^{\perp}$ and $\boldsymbol{v}_m^{\perp}(\Theta)$ are the normal vector for the $\boldsymbol{s}_n$ and $\boldsymbol{v}_m(\Theta)$ respectively.

To reduce the computation cost, we use same σ^2 for Eq. 13 and Eq. 15.

In addition to the above two fitting terms, we also have the **third term** in Eq. 12 that penalize big changes of the poses for neighoring frames. This term is necessary to preserve temporal consistency and smooth motion transition in a sequence.

$$E_r(\Delta\Theta) = \sum ||\Delta\Theta|| \quad (16)$$

Another important constraint we have to take into consideration while performing motion retargeting is the **joint limitation**, as the joint limitation for robots usually differs from the human. This constraint can be handled naturally in our framework by specifying the lower (θ_{min}) and upper bound (θ_{max}) for each joint while performing the optimization for Eq. 12. This can be seen as the hard constraint of the cost function. So the final energy function can be represented as,

$$\begin{aligned} E = E(\Theta, \sigma^2) + \lambda_{\perp} E_{\perp}(\Theta, \sigma^2) + \lambda_r E_r(\Delta\Theta) \\ \Theta \in [\Theta_{min}, \Theta_{max}] \end{aligned} \quad (17)$$

We present two examples in Fig. 4 showing the deformation results after minimizing the energy function Eq. 17. The left columns is the *HUMROB* model in initial poses overlaid with captured 3D mesh data, and the right column is the deformed *HUMROB* template achieved by our method.

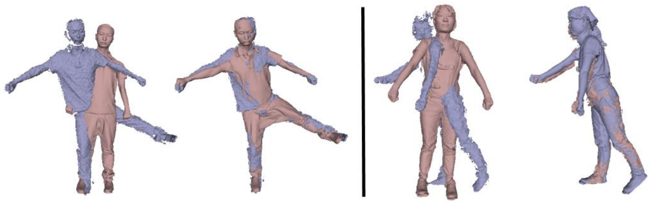


Fig. 4: Deformed HUMROB model results after applying our framework.

2) *Robot stability:* In this paper we assume that the robot is motionless as it is driven at a very low motor speed. This indicates that it only experiences gravitational forces. Therefore instead of using the complex ZMP criteria and we turn to use floor projection of center of mass ($FCoM$) criteria to ensure the stability of the robot. Next we will show how to integrate this stability constraint into our framework.

Suppose the robot has L links or bones and the vectors $\boldsymbol{p}_l$ are the vectors points from the CoM of each individual bones to coordinate origin. The CoM for the whole body denoted as $\boldsymbol{p}_{CoM}$ is computed as,

$$\boldsymbol{p}_{CoM} = \frac{\sum_{l=1}^{L} m_l \boldsymbol{p}_l}{\sum_{l=1}^{L} m_l}, \quad (18)$$

where m_l is the mass of each bone.

The floor projection of the CoM ($\boldsymbol{p}_{FCoM}$) is calculated by taking x and y component of vector $\boldsymbol{p}_{CoM}$ as its x and y component and setting its z component to be zero.

The main criteria is: the motionless humanoid should be able to maintain its balance if $\boldsymbol{p}_{FCoM}$ is in the support polygon (SP), which is formed by the convex hull about the floor support points. In this paper, we use the inscribed circle of SP as the more strict support polygon, denoted as C_SP. We introduce another term E_b express the robot stability, which is computed as,

$$E_b = \begin{cases} 0 & if \boldsymbol{p}_{FCoM} \quad inside \quad C\_SP \\ \infty & if \boldsymbol{p}_{FCoM} \quad outside \quad SP \\ \boldsymbol{o} - \boldsymbol{p}_{FCoM} & otherwise \end{cases} \quad (19)$$

where $\boldsymbol{o}$ is center of C_SP.

Therefore, for each frame we will first find the optimal pose configuration by minimizing Eq. 17. Then we can compute E_b which reflects the current stability status. If E_b equals zero, it means we can safely apply the current pose into the robot with stability already maintained. However, if E_b equals ∞, we will give the opposite offset (0.3 rad) to the hips' pitch/roll angles. Otherwise, we will record the $\overrightarrow{op}_t = \boldsymbol{o} - \boldsymbol{p}_{FCoM}$ for the current frame t. Suppose for the the next frame $t+1$, we have got the vector $\overrightarrow{op}_{t+1}$. Then if $cos(\overrightarrow{op}_t, \overrightarrow{op}_{t+1}) > cos(30°)$ & $||\overrightarrow{op}_{t+1}|| > ||\overrightarrow{op}_t||$, which indicates that the stability violation will probably occur in successive frame, in this case, we will give an opposite offset (0.05 rad) to the hips' pitch/roll angles.

V. Implementation

A. Data Preprocessing

We capture the depth data of human subject using Microsoft Kinect V2. The depth data has the resolution at 512×424 and the depth stream is captured at 30fps. We assume that depth sensor is calibrated and the region of interest, namely the human subject, can been roughly segmented with background subtraction.

The parameters (Θ) are all coded as rad in order to keep consistent with the real robot. The algorithm is implemented in Matlab on an ordinary PC with Intel i7 3.6GHz processor and 16GB RAM and it takes about five seconds for each frame. The pose transferred to real robot is coded with Python through Choregraphe™.

B. Implementation details

Alg. 1 shows how to minimize the energy function Eq. 17 to get the pose configurations, namely the parameters Θ and σ^2. First, we adopt a linearized surface deformation strategy to simply the constrained nonlinear cost function into a linear one [26]. But the normal fitting term Eq. 15 cannot be linearized, so it is neglected in this stage. The computed linear solution can then be taken as the initialization for the over-all nonlinear optimization, for which we choose to use trust-region-reflective algorithm to find a optimal solution for the nonlinear function Eq. 17.

In the Linear solver part, we only consider position fitting (Eq. 13), then the the posteriors p_{mn} is computed as,

$$p_{mn} = \frac{\exp(-\frac{||\boldsymbol{s}_n - \boldsymbol{v}_m^{old}(\Theta)||^2}{2(\sigma^{old})^2})}{\sum_{i=1}^{M}\exp(-\frac{||\boldsymbol{s}_n - \boldsymbol{v}_i^{old}(\Theta)||^2}{2(\sigma^{old})^2}) + c}, \tag{20}$$

In the Nonlinear solver part, the vertex and normal are considered together (Eq. 13 and Eq. 15), so the posteriors p_{mn} is computed as,

$$p_{mn} = \frac{\exp(-\frac{||\boldsymbol{s}_n - \boldsymbol{v}_m^{old}(\Theta)||^2 + \lambda_f||\boldsymbol{s}_n^{\perp} - \boldsymbol{v}_m^{\perp old}(\Theta)||^2}{2(\sigma^{old})^2})}{\sum_{i=1}^{M}\exp(-\frac{||\boldsymbol{s}_n - \boldsymbol{v}_i^{old}(\Theta) + \lambda_f||\boldsymbol{s}_n^{\perp} - \boldsymbol{v}_m^{\perp old}(\Theta)||^2}{2(\sigma^{old})^2}) + c}, \tag{21}$$

And c in Eq. 20 and Eq. 21 can be computed by Eq. 4.

```
initial pose: linear third order auto-regression
begin Step One: Linear solver for initialization
    while Pose not converged do
        E-step: Compute posteriors via Eq. 20.
        M-Step:
          • Minimize the linearized cost function of
            Eq. 13 for $(\Theta, \sigma^2)$
          • Update template vertices via Eq. 14
          • parameters are adjusted by enforcing
            robot stability
    end
end
begin Step Two: Nonlinear solver for global solution
    E-step: Compute posteriors via Eq. 21.
    M-Step:
      • Minimize Eq. 17 using trust-region-reflective
        algorithm
      • Apply stability to adjust the parameters
end
```

Algorithm 1: The parameters optimization procedure

Then, as shown in Alg. 1, the parameters are adjusted in every M-step to satisfy the stability constraints as described in section IV-B.2.

Finally, in order to speed up the convergence speed and reduce the computation cost, two strategies are employed. First, we perform sub-sampling for both the *HUMROB* template and the captured point cloud. We have tested different numbers of sampled points. Without losing key components and good performance, we identified the template and point maps sample number should be 5000 and 2500 respectively. We found these can achieve a good balance. The other strategy is that we adopt the poses of previous frames with a third-order linear auto-regression model as a prediction for the current pose.

C. Parameter settings

In Eq. 13 and Eq. 15, the initial $\sigma^2 = 0.02^2$. In Eq. 21, $\lambda_f = 0.5$. In Eq. 17, $\lambda_{\perp} = 0.5$ and $\lambda_r = 1000$. In Alg. 1, the pose converge criteria is that iteration stops when the maximum movement for all joints is less than $0.002 rad$ given that the precision for the real robot sensor is $0.1°$.

These values are tuned empirically and remained fixed for all the experiments.

VI. Experiments

A. Robot test

We have tested our method on NAO robot that is manufactured by Softbank. The robot is 58 cm tall and weighs 5.2 kg. The robot have 23 DoFs in total for whole body (exclude the open/close hands DoFs), while the generic human skeleton has 47 DoFs. The specific robot configuration has been enforced by embedding it into the *HUMROB* model.

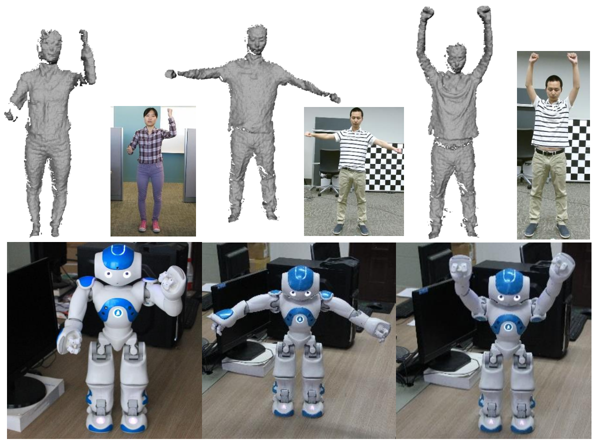
The framework is validated on human subjects both male and female, with subjects performing various and even some extreme motions. In Fig. 5 shows our retargeting results of several different kinds of postures.

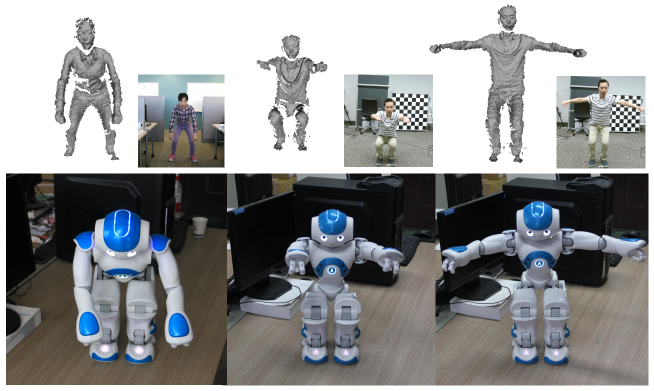
As shown in Fig. 5a, the robot can mimic the hands up postures performed by the human subjects successfully. In Fig. 5b, bow and squat postures can also be executed by the robot resembling to the human. These postures mainly focus on upper/lower body posture with limited stability constraints.

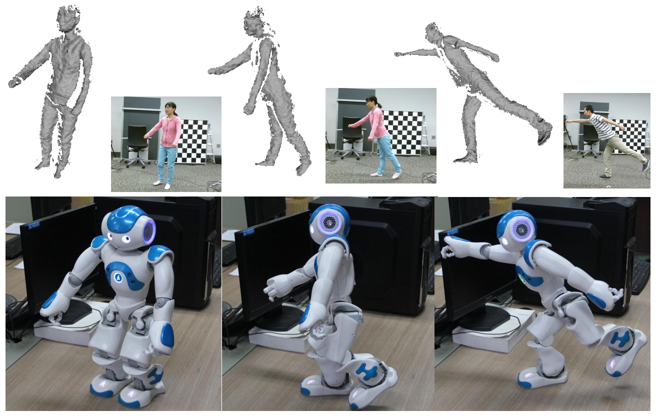
We present more results in Fig. 5c and Fig. 5d with human subjects leaning forward and kicking. The stability needs to be taken into consideration under these circumstances, especially in one foot support, stability has the highest priority protecting the robot from falling down. Though some postures are not executed exactly as the human subjects, they still share great similarity globally.

B. Stability

The stability check described in section IV-B.2 plays an important role in the motion retargeting. While we try to make the robot to perform the poses as similar as possible to the human subject, we have to take its stability into consideration as the robot may fall down in some poses. We present two cases in Fig. 6, that shows the retargeting results with and without our stability constraints. The middle columns are simulation results without stability requirements, which are not subjected to gravity force, so the robot acts just like the human subject. However, if we apply the same pose onto the real robot, it will fall down. After enforcing the stability constraints, we will get the results as shown in the right columns. We try to achieve the most similar poses while maintaining its stability.


(a) Hands up


(b) Bow and Squat


(c) Rotation and Lean forward

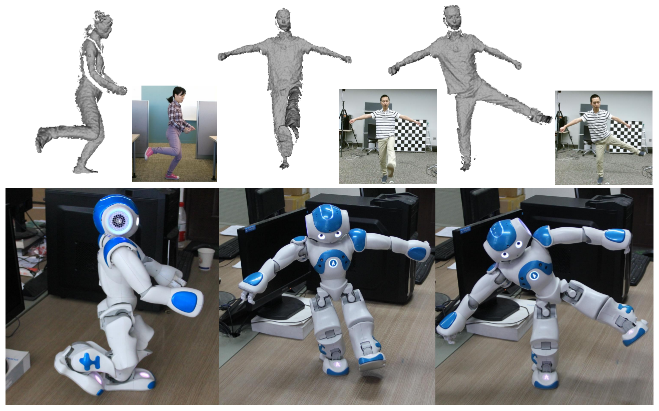

(d) Kick

Fig. 5: Results in different kinds of postures. For each case, the first row shows the original mesh and image, and the second row shows the robot executing the pose computed with our proposed approach.

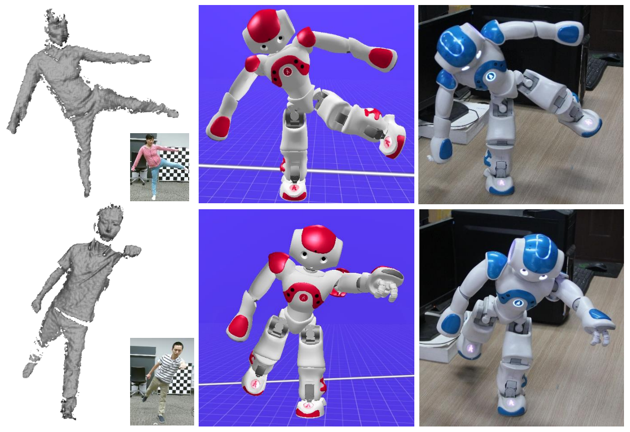
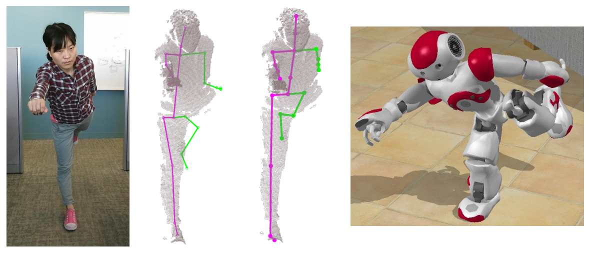


Fig. 6: The effect of robot stability. The left column is original mesh, the middle column is simulation result without gravity and stability constraints, the right column is the retargeting on real robot with stability applied.


(a) Lean forward

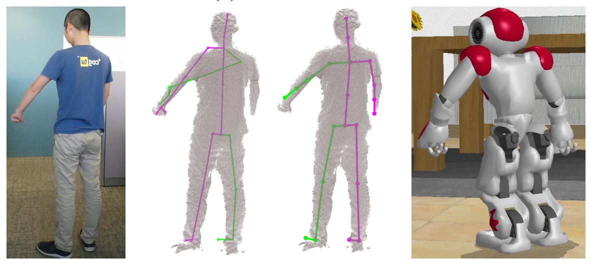

(b) Rotation

Fig. 7: Visual comparisons of our results with SDK on two cases. In each case, the left one is the captured image, the middle left is the result from the SDK while the middle right one is result using our framework, the right one is the simulation result for motion retargeting.

In the real humanoid test, we also restrict that at least one foot fully touches the ground. Besides, the support leg of the robot needs to bend a little if it almost reaches the maximum motor torque, as shown in the third pose in Fig. 5d.

C. Comparison

We have done some qualitative comparisons between our methods and Kinect SDK as shown in Fig. 7.

The skeletons are colored with green and magenta representing the left and right part of the body, respectively. In Fig. 7a, the human subject leans forwards with one of her legs severely occluded. The skeleton we get from Kinect SDK is clearly wrong for the leg, while we can still get more reasonable result. Fig. 7b shows that when the subject turns round, the SDK cannot differentiate the front and the back of the human body which cause the flipping of left and right sides of the body. However, since we take the depth sequence into consideration and penalize abrupt changes in successive frames rather than tracking frame by frame, we are able to handle these cases.

VII. Conclusion

In this paper, we propose a novel generative framework for human-robot motion retargeting with motion tracking and retargeting computed in a single formulation. A parametric *HUMROB* template model is built with the robot configurations embedded, from which we can formulate the cost function to maintain the similarity between deformed template model and captured data from depth sensor. The joint limitation of the robots and stability constraints are enforced in the framework. No pre-defined joints mapping between robot and human subject is needed, instead we can take the similarity for both joint angles and the end position simultaneously into consideration. We have validated our method on both simulated and real robots to verify its ability of motion retargeting for various poses.

Limitations: 1) For some self-interactive motions like his/her hand touching the head, the interaction may not be maintained by the robot since we haven't enforced interaction constraints explicitly in our formulation. 2) Due to non-optimized Matlab code and motionless assumption, the whole process is not real-time. These are two of our major works to be improved in the future.

ACKNOWLEDGMENT

This work is partially supported by the National Natural Science Foundation of China (No. 61603302, 51375390, 61332017, 61170324, 61572020), US National Science Foundation grants (IIS-1231545, IIP-1543172), Key Industrial Innovation Chain of Shaanxi Province Industrial Area (2016KTZDGY06-01, 2015KTZDGY04-01), and the "111 project" of China (No. B13044). Xinxin Zuo and Ruigang Yang are the co-corresponding authors for this paper.

References

[1] M. J. Gielniak, C. K. Liu, and A. L. Thomaz, "Generating human-like motion for robots," *Int. J. Robot. Res.*, vol. 32, no. 11, pp. 1275–1301, 2013.

[2] S. Guo, R. Southern, J. Chang, D. Greer, and J. Zhang, "Adaptive motion synthesis for virtual characters: a survey," *The Visual Computer*, vol. 31, no. 5, pp. 497–512, 2015.

[3] O. E. Ramos, N. Mansard, O. Stasse, C. Benazeth, S. Hak, and L. Saab, "Dancing humanoid robots: Systematic use of osid to compute dynamically consistent movements following a motion capture pattern," *IEEE Robot. Autom. Mag.*, vol. 22, no. 4, pp. 16–26, 2015.

[4] L. P. Poubel, S. Sakka, D. Cehajic, and D. Creusot, "Support changes during online human motion imitation by a humanoid robot using task specification," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2014, pp. 1782–1787.

[5] J. P. BANDERA, J. A. RODRÍGUEZ, L. MOLINA-TANCO, and A. BANDERA, "A survey of vision-based architectures for robot learning by imitation," *Int. J. of Hum. Robot.*, vol. 9, no. 1, pp. 1–40, 2012.

[6] B. Dariush, Y. Zhu, A. Arumbakkam, and K. Fujimura, "Constrained closed loop inverse kinematics," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2010, pp. 2499–2506.

[7] G. B. Hammam, P. M. Wensing, B. Dariush, and D. Orin, "Kinodynamically consistent motion retargeting for humanoids," *International Journal of Humanoid Robotics*, vol. 12, no. 04, p. 1550017, 2015.

[8] Y. Zhu, B. Dariush, and K. Fujimura, "Kinematic self retargeting: A framework for human pose estimation," *Computer vision and image understanding*, vol. 114, no. 12, pp. 1362–1375, 2010.

[9] J. Koenemann, F. Burget, and M. Bennewitz, "Real-time imitation of human whole-body motions by humanoids," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2014, pp. 2806–2812.

[10] M. Lee and I. Cohen, "Proposal maps driven mcmc for estimating human body pose in static images," in *Proc. IEEE CVPR*, 2004, pp. II–334.

[11] T. Tosun, R. Mead, and R. Stengel, "A general method for kinematic retargeting: Adapting poses between humans and robots," in *Proc. ASME 2014 Int. Mech. Eng. Congress Expo.*, 2014.

[12] Y. Ou, J. Hu, Z. Wang, Y. Fu, X. Wu, and X. Li, "A real-time human imitation system using kinect," *International Journal of Social Robotics*, vol. 7, no. 5, pp. 587–600, 2015.

[13] C. Plagemann, V. Ganapathi, D. Koller, and S. Thrun, "Real-time identification and localization of body parts from depth images," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2010, pp. 3108–3113.

[14] J. Shotton, A. Fitzgibbon, M. Cook, T. Sharp, M. Finocchio, R. Moore, A. Kipman, and A. Blake, "Real-time human pose recognition in parts from single depth images," in *Proc. IEEE CVPR*, 2011, pp. 1297–1304.

[15] M. Ye and R. Yang, "Real-time simultaneous pose and shape estimation for articulated objects using a single depth camera," in *Proc. IEEE CVPR*, 2014, pp. 2353–2360.

[16] Y. Yang, V. Ivan, and S. Vijayakumar, "Real-time motion adaptation using relative distance space representation," in *Proc. IEEE ICAR*, 2015, pp. 21–27.

[17] Y. Seol, C. O'Sullivan, and J. Lee, "Creature features: online motion puppetry for non-human characters," in *Proc. ACM SCA*, 2013, pp. 213–221.

[18] K. Yamane, Y. Ariki, and J. Hodgins, "Animating non-humanoid characters with human motion data," in *Proc. ACM SCA*, 2010, pp. 169–178.

[19] A. Myronenko and X. Song, "Point set registration: Coherent point drift," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, no. 12, pp. 2262–2275, 2010.

[20] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the em algorithm," *Journal of the royal statistical society*, pp. 1–38, 1977.

[21] R. M. Murray, Z. Li, and S. S. Sastry, *A Mathematical Introduction to Robotic Manipulation*. Boca Raton, FL, USA: CRC Press, Inc., 1994.

[22] J. P. Lewis, M. Cordner, and N. Fong, "Pose space deformation: a unified approach to shape interpolation and skeleton-driven deformation," in *Proc. ACM SIGGRAPH*, 2000, pp. 165–172.

[23] J. Tong, J. Zhou, L. Liu, Z. Pan, and H. Yan, "Scanning 3d full human bodies using kinects," *IEEE Trans. Vis. Comput. Graph*, vol. 18, no. 4, pp. 643–650, 2012.

[24] Q. Zhang, B. Fu, M. Ye, and R. Yang, "Quality dynamic human body modeling using a single low-cost depth camera," in *Proc. IEEE CVPR*, 2014, pp. 676–683.

[25] T. Helten, A. Baak, G. Bharaj, M. Muller, H. Seidel, and C. Theobalt, "Personalization and evaluation of a real-time depth-based full body tracker," in *Proc. IEEE 3DV*, 2013, pp. 279–286.

[26] M. Ye, Y. Shen, C. Du, Z. Pan, and R. Yang, "Real-time simultaneous pose and shape estimation for articulated objects using a single depth camera," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 38, no. 8, pp. 1517–1532, 2016.