Motion Capture Retargeting: Preserving Surface Spatial Relationship

Rodolfo L. Tonoli [*]

Department of Computer Engineering and Industrial Automation
School of Electrical and Computer Engineering
University of Campinas

Abstract

Motion Capture is used to record the movements of a real actor and animate virtual characters. It generates intrinsically realistic animations, since it tracks the motion of a real person. But the process to transfer the motion to a character is not straigthforward since the actor and the character may have distinct body proportions and distinct skeleton topology. The present work focus on retargeting the motion from Motion Capture to a 3D character while preserving the spatial relationship betweeen extremity joints, the hands and the feet, and the body surface. This process retain the semantic information information of movements in which the hands and the feet interact with the surface of the body, as holding the hands in front of the eyes or mouth. The Motion Retargeting process described computes new positions of the extremity joints regarding surface components of the body and adapts the pose of the virtual character in the posistions computed using Inverse Kinematics. The motion can be transfered to topologically different skeletons. The results shows that the resulting motion is being adapted accordingly to the body surface and it is as smooth as the original motion capture data.

Keywords: Motion Capture, Motion Retargeting, 3D Animation, Virtual Agents.

Index Terms: Computing methodologies [Computer graphics]: Animation—Motion capture

1 Introduction

Toy Story was released in 1995 as the first full length film produced entirely through computer animation techniques [3]. Besides Pixar's production, several applications employ digital animations to entertain, convey information, educate, among others. Some exploits the use of virtual human models to make the human-computer interatction more natural and accessible. Talita, the virtual human of the TAS project [2], is a signing avatar that communicates with deaf students using the Brazilian Sign Language. The goal of the TAS project is to help the deaf and hard of hearing to access information and in their educational process.

Humans use not only the voice and facial expression cues to understand intentions, motives and wills, but also the movements of our body and limbs may carry semantic information on the emotion that one is triyng to express. The location, orientation and speed of the hand are some of the parameters that characterizes a sign language gesture, slightly discrepancies will express different messages or even make the gesture unrecognizable. Therefore, an accurate motion representation by the virtual agent should be of great concern.

[*]e-mail: rltonoli@gmail.com

Keyframing is a technique to animate an avatar, the animator adjusts crucial poses of the 3D character on dispersed frames and an algorithm fills the gap between two poses by interpolating them over time, which creates the impression of the desired motion. Another digital animation technique is Motion Capture (MoCap) that aims to lessen the time-consuming and tedious work of keyframing animations. MoCap systems registers an actor performing the desired motion to animate a 3D character. Optical Motion Capture systems, for example, uses a set of cameras to track the position over time of reflective markers placed on the body surface of the performer. The motion is then transferred to the avatar, the motion retargeting process, without the need to create all the poses along the action.

The motion retargeting, however, can cause ill-conditioned poses when the body proportions of the performer and the avatar are different. The motion retargeting of an actor with long arms covering his ears, as an example, to a 3D character with short arms results in a unrecognizable action, since the hands of the avatar will not reach its ears. In this case, the animator must inspect the animation, identify the irregular artifacts and correct them with keyframing, adjusting the avatar pose to the desired one. Some motion retargeting algorithms aims to avoid such artifacts to further reduce the inspection and adjustments needed by the animator.

Furtheremore, the standard approach to animate 3D characters is the skeleton animation. The skeleton, a hierarchical structure of joints and bones, stores the actual motion data and the surface of the character is deformed accordingly to the rotation and translation of the skeleton. Often MoCap systems, game engines, 3D modeling and animation software built the skeleton with different specifications(cite Kitagawa), i.e., distinct topologies of the hierarchical structure. This complicates the motion retargeting process because the joints and bones of one skeleton may not exist in another or be represented in another way.

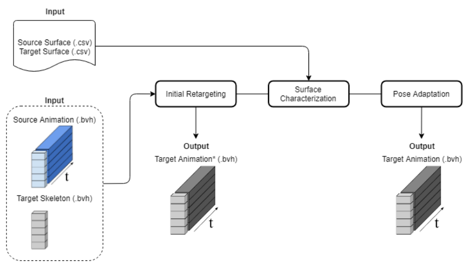

Figure 1: Workflow

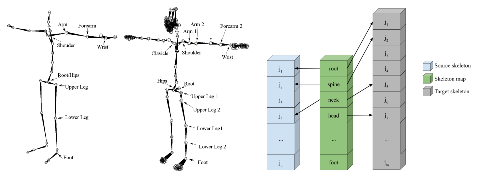

Figure 2: Skeleton Map

2 Method

2.1 Initial Retargeting

2.1.1 Skeleton Animation

This Section describes the Initial Motion Retargeting process, which receives the source animation and the target skeleton as inputs, both BVH files. The output is an animation of the target skeleton performing the movements of the source skeleton that can be exported as a BVH file. The success of the process does not depend on matching topologies of the skeletons, but the first pose of the source animation and the target skeleton pose must be as close as possible to the T-Pose. The output of the Initial Retargeting is the target skeleton animation. The target animation does not exploit yet the "surface-awareness" adjustments that the methods described in the subsequent sections provide, but it can already be imported in game engines or animation softwares to animate a 3D character.

To animate a skeleton, its joints must rotate and translate along the frames, conveying the impression of motion. The hierarchical representation of the skeleton allows an all-around description of the motion by only preserving the rotational and translational values of a joint regarding its parent joint, i.e., local rotations and translations. The local transform matrix M is the combination of both rotation R and translation matrix T. Given a joint n in the skeleton, its global transform matrix M^n_{global} is the combination of the local transform matrices of all the joints above the hierarchy:

$$M^n_{global} = \prod_{i=0}^{n} M^i_{local} \tag{1}$$

2.1.2 Skeleton Mapping

Figure 2 presents two approaches to represent a virtual human skeleton. Although both can animate a humanoid-shaped model, transferring motions between the skeletons is not straightforward. A correspondence between joints of the skeletons is mandatory to identify which joints from the target skeleton should mimic the motion from the source skeleton. It is possible to infer a correspondence using the joints' name automatically, but note that they may not be placed at the same point in the respective skeleton, as the shoulders in the figure. Therefore, the user can provide a correspondence between skeletons to properly map joints and to perform the motion retargeting [4].

In this work, it is required that both target and source skeletons have at least the set of joints: one hips joint, three spine joints, neck and head joints, and right and left joints for the shoulders, elbows, wrists, femurs, knees and feet. The joints referenced will be the only ones used in the Initial Retargeting proccess.

2.1.3 Bones Alignment

Bones Alignment enforces that the vector of a mapped joint from the target skeleton pointing towards its child joint has the same direction of the correspondent vector from the source skeleton. Then, for every frame, we apply the same transform from the source skeleton joint to the correspondent target joint.

The rotation matrix R_A to align the bone vector of the target skeleton onto the bone vector of the source skeleton is calculated and applied to the global rotation of the target skeleton joint R^n_G:

$$R^n_{NG} = R_A R^n_G \tag{2}$$

where R^n_{NG} is the new global rotation. Then we recover the local rotation of the joint R^n_L, the new global orientation is multiplied by its parent inverse rotation matrix.

$$R^n_{NG} = \left(\prod_{i=0}^{n-1} R^i_L \right) R^n_L \tag{3}$$

$$R^n_{NG} = R^{n-1}_G R^n_L \tag{4}$$

$$R^n_L = (R^{n-1}_G)^{-1} R^n_{NG} \tag{5}$$

In the following frames, the rotation of a joint in the source animation, from the last frame to the current one, replaces the rotation R_A to align the correspondent joint in the target skeleton.

We compute the ratio of the heights of the root joint from the target and source skeletons in the first frame. The position of the root joint in the target skeleton is adapted using Equation 6.

$$\mathbf{p}(t)_{tgt} = \mathbf{p}(t)_{src} ratio \tag{6}$$

2.2 Surface Calibration

Molla [5] defined a set of points to characterize the surface of the 3D model's and performer's body, the yellow dots in Figure 3. The points are connected throughtriangles to create a mesh that represents the body surface. The points on the surfaceof the character and the performer need to be collected as close to the relative positionas possible, since the triangle mesh differences between the characters must represent differences of the characters' surface, thus sampling surface points from different placesintroduces noise to the analysis. The thickness of the limbs are also registered and later they are represented by capsules to avoid self-penetration in the animation.

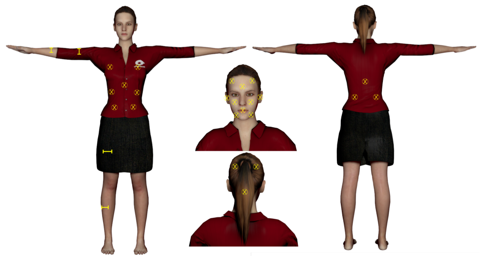
Figure 3: Surface points sampled in the 3D character for the Surface Calibration.

The user obtains the surface points and radius of each limb of the character during its creation or through a 3D modeling software.

2.2.1 Surface Motion Estimation

Collecting the position of the surface points is not enough because the performer may walk around, jump or simply twist his torso. In those cases, the sampled points do not represent the current surface position, since we do not keep track of these points. To include the surface deformation - translation and twist effects - we attach each calibration point representing the character mesh (yellow dots in Figure) to the nearest mapped joint. Thus, the point behaves as a child of the mapped joint and the transformations from every joint in the hierarchy is also applied to the point.

2.3 Computing Egocentric Coordinates

We compute the egocentric coordinates of each limb joint in respect to each triangle in the surface mesh and for each capsule limb. Molla computes the projection of the joint's position $\mathbf{p}$ on the m triangles of the mesh, the projected point $\mathbf{x}$ is called the reference point, and the displacement vector $\mathbf{v}$ is the distance of the joint's position to the reference point. Then, the position of joint n regarding the surface component i is expressed as the equation:

$$\mathbf{p}_n = \mathbf{x}_i + \mathbf{v}_i \tag{7}$$

The position is expressed likewise for the limbs, but the reference point is the intersection between the capsule surface and the line that passes through the center of the capsule and the joint's position. Then, combining all the surface components, limbs and meshes, the previous equation becomes:

$$\mathbf{p}_n = \sum_{i=1}^{m} \hat{\lambda}_i(\mathbf{x}_i + \mathbf{v}_i) \tag{8}$$

The importance factor λ of a surface component is metric that encodes the proximity and orthogonality between the component and the joint's position. The goal of the importance factor is allow that surface components that are near and more perpendicular to the joint have a higher contribution on the joint's position calculation.

Furthermore, a small distance between the joint and the surface may indicate that a interaction is occurring, covering the eyes with the hand as an example. In this case, we desire that the weights of the hand position that a surface component in head express excel the weights of the torso surface component, in order to preserve the interaction between the head and not with the torso.

The importance factor of a surface component is the combination of the proximity λ_p and the orthogonality $\lambda_\perp$ properties:

$$\lambda = \lambda_p \lambda_\perp \text{ , } with \;\; \lambda_p = \frac{1}{||\mathbf{v}||} \;\; and \;\; \lambda_\perp = cos(\alpha) \tag{9}$$

where α is the angle between the displacement vector $\mathbf{v}$ and the surface component normal. Finally, $\hat{\lambda}_i$ is computed as

$$\hat{\lambda}_i = \frac{\lambda_i}{\sum_{j=1}^{m} \lambda_j} \tag{10}$$

Using the target character surface information to compute the reference points and the displacement vectors from the source skeleton, and reversing Equation, we obtain the position for a target skeleton joint with the same distance from its surface as the source skeleton distance from its respective surface. But differences in the body proportions and bones length between the skeletons still were not taken into account, which will result in odd-looking poses or in a position impossible to reach.

2.3.1 Kinematic Path Normalization

Molla proposes the Kinematic Path Normalization to adjust the joints' position according to the length of the bones in the skeleton. The kinematic path is the route through joints and bone segments from a reference point to a limb joint. Figure 4 represents the kinematic path for the right hand to the root.

Note that the goal of the Kinematic Path Normalization is to adjust the position of the joint accordingly to the length of the bone segments. Thus, since the source and target skeleton may have different topologies, we consider a set of bone segments that can be mapped between skeleton with distinct levels of details. This set of bones represent a basic skeleton with the bones: spine, head, and right and left clavicle, arm, forearm, femur, thight and shin.

Given the path along the skeleton from the joint being evaluated until the nearest joint of the surface component or until the extremity joint for limbs capsules, we compute the displacement vector as

$$\mathbf{v} = -\mathbf{x}_{j0} + \sum_{i=1}^{n} \mathbf{s}_i \tag{11}$$

Molla computes the contribution of each bone segment $\mathbf{s}_i$ in the kinematic path to the position of the joint relative to the surface, that is, the displacement vector $\mathbf{v}$, based on the projection of each bone vector onto the displacement vector by

$$cos(\alpha_i) = \frac{\mathbf{v}}{||\mathbf{v}||} \cdot \frac{\mathbf{s}_i}{||\mathbf{s}_i||} \tag{12}$$

The cosines of every bone in the kinematic path are kept as $C_i = \{|cos(\alpha_1)|, |cos(\alpha_2)|, ..., |cos(\alpha_n)|\}$ for a kinematic path with n bone segments. To adapt the joint position in the target skeleton, the displacement vector is normalized as

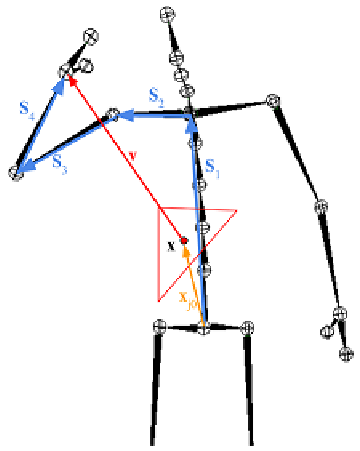

Figure 4: Kinematic Path

$$\hat{\mathbf{v}} = \frac{\mathbf{v}}{\tau}, \textit{ where } \tau = \sum_{i=0}^{n} ||\mathbf{s}_i|| \, |cos(\alpha_i)| \tag{13}$$

2.3.2 Summary

Given a joint j and a surface component i, we store the set of parametes: $\mathbf{e}_{j,i} = (\hat{\lambda}_i, \hat{\mathbf{v}}, C_i)$. The egocentric coordinates of a joint E_j for a given frame considering all m surface components is given by

$$E_j = \{e_{j,1}, e_{j,2}, ..., e_{j,m}\} \tag{14}$$

2.4 Pose Adaptation

To adapt the pose of the target skeleton enforcing the same surface spatial relationship as the souce motion, we denormalize the egocentric coordinates with the parameters regarding the body proportions and bone length of the target skeleton and surface. By inverting Equation 7, we compute the correct joint position in the target skeleton of a extremity joint, then we use Inverse Kinematics so that the skeleton reaches the new position.

First, we compute the reference point of each mesh component of the target character using the surface estimation from Section Surface Estimation. Then, we use the length of the bone segments to calculate the importance factor τ for the target skeleton and multiply it by each normalized displacement vector $\hat{\mathbf{v}}_i$. Finally, the new position of the joint is given by Equation 8.

2.4.1 Inverse Kinematics

We use the Jacobian Transpose method for the Inverse Kinematics (IK) to compute the new pose of the avatar [1]. The kinematic paths fed to the IK algorithm are composed of all joints from the beginning of the limb to the limb extremity: the shoulder to the hand and the upper leg to the foot. This allows a separete IK computing for each limb and avoids dragging the spine and the hips in order to reach the target position.

3 Results

Figure 5 shows the pose of the source and target skeletons of a motion capture take in which the actor is touching its

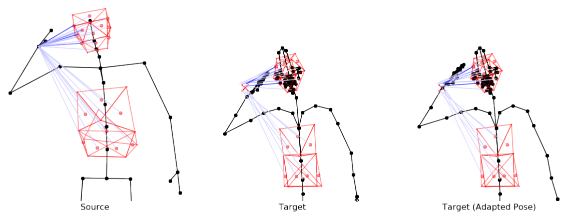

Figure 5: Results.

right eyebrow with his fingers. In the middle, the target animation is the result of the bones alignment, where the red X indicates the correct position of the right hand joint computed through the egocentric coordinates. The Inverse Kinematis receives the pose and the correct position and computes the new pose of the target skeleton, on the right.

Darker colors of blue represent a greater importance of the displacement vectors. As expected from a movement close to the head, mesh componets of the head are contributing more than those of the body. Furthermore, the triangles with normal vectors pointing in the same direction as the displacement vectors are also receiving greater weights, which indicates that the orthogonality factor exposes possible interaction of the joints with the surface.

4 Conclusion

We applied the method described by Eray Molla to ensure the interaction of the extremity joints with the body surface when retargeting motion from Motion Capture data. With the body surfaces calibrated, the algorithm computes a new pose for the target skeleton and uses Inverse Kinematics to adapt the skeleton into the desired pose. We found that the method is able to improve the target animation when the hands are interacting with the body surface.

In future works, we wish to conduct a perpectual evaluation to assess the resemblance between the target animations, adpated and non-adapted, and the video of the Motion Capture actor performing the same movement. Furthermore, towards sign language avatars, the work could be extended to the fingers adaptation, since hand configuration is a key aspect on sign characterization.

References

[1] S. R. Buss. Introduction to inverse kinematics with jacobian transpose, pseudoinverse and damped least squares methods. *IEEE Journal of Robotics and Automation*, 17(1-19):16, 2004.

[2] J. M. De Martino, I. R. Silva, C. Z. Bolognini, P. D. P. Costa, K. M. O. Kumada, L. C. Coradine, P. H. d. S. Brito, W. M. do Amaral, Â. B. Benetti, E. T. Poeta, L. M. G. Angare, C. M. Ferreira, and D. F. De Conti. Signing avatars: making education more inclusive. *Universal Access in the Information Society*, 16(3):793–808, 2017. doi: 10.1007/s10209-016-0504-x

[3] M. Henne, H. Hickel, E. Johnson, and S. Konishi. The making of toy story [computer animation]. In *COMPCON'96. Technologies for the Information Superhighway Digest of Papers*, pp. 463–468. IEEE, 1996.

[4] M.-K. Hsieh, B.-Y. Chen, and M. Ouhyoung. Motion retargeting and transition in different articulated figures. In

Ninth International Conference on Computer Aided Design and Computer Graphics (CAD-CG'05), pp. 6–pp. IEEE, 2005.

[5] E. Molla, H. G. Debarba, and R. Boulic. Egocentric mapping of body surface constraints. *IEEE transactions on visualization and computer graphics*, 24(7):2089–2102, 2018. doi: 10.1109/TVCG.2017.2708083