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

1 Abstract

Bla bla bla

2 Introduction

Toy Story was released in 1995 as the first full length film produced entirely through computer animation techniques [2]. 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 [1], 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 uses 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.

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.

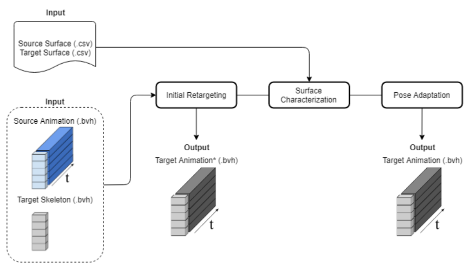
Fruthermore, the standard approach to animate 3D characters is the skeleton animation. The skeleton, a hierarchical structure of joints and bones, stores the actual motion data and the surface of the character is deformed accordingly to the rotation and translation of the skeleton. Often MoCap systems, 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.

*e-mail: rltonoli@gmail.com

3 Related Works

(em construção)

4 Workflow



Workflow

5 Initial Retargeting

5.1 Skeleton Animation

This chapter 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:

$$M = TR = \begin{bmatrix} R_{3x3} & T \\ 0 & 1 \end{bmatrix} \tag{1}$$

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. The position of the joint is computed through the global transform matrix.

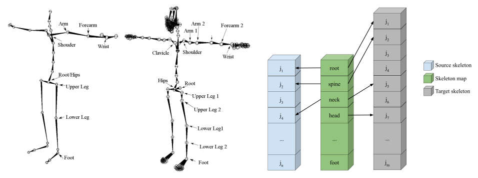
$$M^n_{global} = \prod_{i=0}^{n} M^i_{local} \tag{2}$$

$$\mathbf{p}_n = M^n_{global}[0,0,0,1]^T \tag{3}$$

```
Preparing plot:
Computing Bones...
100 frames done.
200 frames done.
300 frames done.
400 frames done.
500 frames done.
600 frames done.
Done!
```

5.2 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 (Detailed in Appendix A) (cite Monzani and Hsieh).



Skeleton Map

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.

5.3 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_{NG}:

$$R^n_{NG} = R_A R^n_G \tag{4}$$

Then, as expected by the BVH file format, we recover the local rotation of the joint R^n_L, the new global orientation is multiplied by its parent inverse rotation matrix.

$$R^n_{NG} = \left(\prod_{i=0}^{n-1} R^i_L \right) R^n_L \tag{5}$$

$$R^n_{NG} = R^{n-1}_G R^n_L \tag{6}$$

$$R^n_L = (R^{n-1}_G)^{-1} R^n_{NG} \tag{7}$$

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.

The source motion of the root joint is decomposed into vertical and horizontal movement that are analyzed separately. The horizontal movement is the projection of the root joint position into the ground, the reason of the decomposition is the use of this projected point as a reference point for the feet (Section Surface Characterization).

The ratio is given by:

$$ratio = \frac{h^{tgt}_{root}}{h^{src}_{root}} \tag{8}$$

where h^{tgt}_{root} and h^{src}_{root} are the heights of the root joint of the target and source skeletons in the first frame. The horizontal and vertical movement of the root, $g(t)$ and $h(t)$ is computed. The translation of the root joint regarding the system origin is equal to these values recombined.

$$\mathbf{g}(t)^{src}_{root} = \mathbf{g}(t)^{tgt}_{root} ratio \quad and \quad \mathbf{h}(t)^{src}_{root} = \mathbf{h}(t)^{tgt}_{root} ratio \tag{9}$$

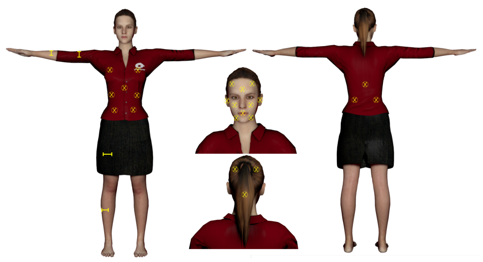
```
Starting Posture Initialization
100 frames done. 8.70 seconds.
200 frames done. 7.43 seconds.
300 frames done. 9.52 seconds.
400 frames done. 5.11 seconds.
500 frames done. 9.88 seconds.
600 frames done. 10.07 seconds.
Done!
```

```
Preparing plot:
Computing Bones...
100 frames done.
200 frames done.
300 frames done.
400 frames done.
500 frames done.
600 frames done.
Done!
```

6 Surface Characterization

6.1 Surface Calibration

Molla defined a set of points to characterize the surface of the 3D model's and performer's body, the yellow dots on Figure 5.1. 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.



Character Surface Calibration

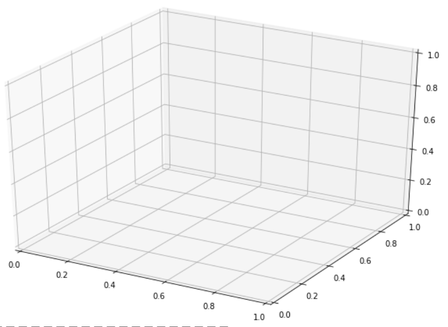
The user obtains the surface points and radius of each limb of the character during its creation or through a 3D modeling software.

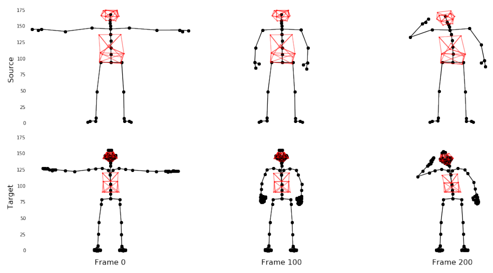
6.2 Surface Motion Estimation

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.

```
Preparing plot:
Computing Surface...
```

```
    ␣
↪--------------------------------------------------------------------------

      AssertionError                              ␣
↪    Traceback (most recent call last)

      <ipython-input-14-2e709cdb796a> in␣
↪<module>
        54     return ani
        55
  ---> 56 _ =␣
↪PlotBVHSurface(sourceAnimation,sourceSurface)


      <ipython-input-14-2e709cdb796a> in␣
↪PlotBVHSurface(animation, surface)
        21         vertices_oneframe = []
        22         for triangle in surface.
↪headmesh + surface.bodymesh:
  ---> 23             edge_0 = triangle[0].
↪getPosition(animation,frame)
        24             edge_1 = triangle[1].
↪getPosition(animation,frame)
        25             edge_2 = triangle[2].
↪getPosition(animation,frame)



      ␣
↪~\Desktop\MotionRetargeting\dev\surface.py␣
↪in getPosition(self, animation, frame)
       210             joint = skeletonmap.
↪getmatchingjoint(self.jointlock, animation)
       211             assert joint, str.
↪format('Could not find attached joint in␣
↪animation: %s' % self.jointlock)
  --> 212             assert self.
↪calibrationLocalTransform.all(), str.
↪format('Calibration Local Transform not␣
↪computed for point %s' % self.name)
       213             globalPointTransform =␣
↪np.dot(joint.getGlobalTransform(frame),␣
↪self.calibrationLocalTransform)
       214             return np.
↪dot(globalPointTransform,[0,0,0,1])[:-1]


      AssertionError: Calibration Local␣
↪Transform not computed for point headRight
```





6.3 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:

$$\mathbf{p}_n = \mathbf{x}_i + \mathbf{v}_i \tag{10}$$

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:

$$\mathbf{p}_n = \sum_{i=1}^{m} \hat{\lambda}_i(\mathbf{x}_i + \mathbf{v}_i) \tag{11}$$

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 the torso.

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

$$\lambda = \lambda_p \lambda_\perp \ , \quad with \quad \lambda_p = \frac{1}{||\mathbf{v}||} \quad and \quad \lambda_\perp = cos(\alpha) \tag{12}$$

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

$$\hat{\lambda}_i = \frac{\lambda_i}{\sum_{j=1}^{m} \lambda_j} \tag{13}$$

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.

6.4 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 BLA represents three different kinematics path for the limb joint right hand: first and second, the path from the reference point of a torso and a head mesh component; and third, the path from the reference point of a limb capsule.

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, the joints and bone segments of the kinematic path are (the mapped joints).

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

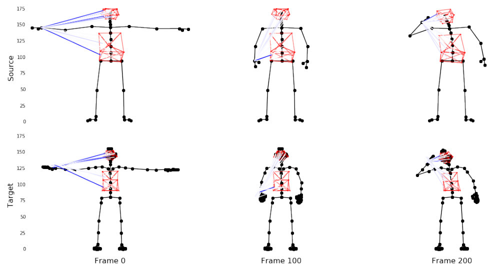
$$\mathbf{v} = -\mathbf{x}_{j0} + \sum_{i=1}^{n} \mathbf{s}_i \tag{14}$$

COSSENOS
TAU

```
100 frames done. 37.05415320396423 seconds.
200 frames done. 38.57540678977966 seconds.
300 frames done. 36.319700717926025 seconds.
400 frames done. 37.326499938964844 seconds.
500 frames done. 28.35040545463562 seconds.
600 frames done. 30.99491047859192 seconds.
Done!
```

```
[39]: array([[ 1.95824617e+00,  1.51431879e+02, -1.
       ↪42545257e+00],
              [-2.81882290e+00,  1.46295469e+02, -3.
       ↪72165332e+00],
              [-2.22049505e+00,  1.41711300e+02, -2.
       ↪78065568e+00],
              [ 6.86468713e+00,  1.46681771e+02, -3.
       ↪82499248e+00],
              [ 6.63866394e+00,  1.41991156e+02, -2.
       ↪88002111e+00],
              [-3.97381013e-01,  1.48955237e+02, -8.
       ↪01542890e-01],
              [ 4.38913877e+00,  1.49148277e+02, -7.
       ↪98438191e-01],
              [ 2.52814435e+00,  1.39010957e+02,  1.
       ↪05570754e+00],
              [ 2.23110510e+00,  1.43525496e+02,  4.
       ↪20662584e-01],
              [ 2.03351158e+00,  1.46671635e+02, -1.
       ↪74528509e-01],
       [ 1.90636139e+00,  1.48043929e+02, -1.
↪65817302e+01],
       [-3.11605565e+00,  1.47734901e+02, -8.
↪87812158e+00],
       [ 7.02217907e+00,  1.48149312e+02, -9.
↪05106180e+00],
       [ 2.33791469e+00,  1.15708645e+02,  2.
↪77855610e+00],
       [-2.78212659e+00,  1.11179571e+02,  3.
↪47673579e+00],
       [ 7.59058080e+00,  1.11460814e+02,  3.
↪51896813e+00],
       [ 2.47053952e+00,  1.06931739e+02,  4.
↪21714783e+00],
       [ 1.82398405e-02,  9.57750210e+01,  4.
↪77758201e+00],
       [ 5.00788379e+00,  9.58553698e+01,  4.
↪84068652e+00],
       [ 2.70856119e+00,  9.35856573e+01, -1.
↪03931548e+01],
       [-5.03238852e+00,  9.57793503e+01, -3.
↪32990512e-01],
       [ 1.01454312e+01,  9.59652674e+01, -4.
↪53048712e-01],
       [ 4.26924639e+01,  1.22854779e+02, -4.
↪17720395e+00],
       [ 1.86414697e+01,  1.23743222e+02, -7.
↪84384488e+00],
       [-1.42481811e+01,  3.18096833e+01, -7.
↪86362476e+00],
       [-1.33948906e+01,  6.72910571e+01, -4.
↪28438000e+00],
       [ 9.49221512e+00,  3.01292512e+01, -7.
↪99556077e+00],
       [ 7.02590242e+00,  6.62089048e+01, -4.
↪56816643e+00]])
```

```
[53]: array([0.00139762, 0.11825547, 0.13162839, 0.
      ↪00134696, 0.00137559,
             0.00148639, 0.00135538, 0.00491221, 0.
      ↪05536959, 0.00143716,
             0.00147203, 0.10882928, 0.00134097, 0.
      ↪00151919, 0.0207191 ,
             0.00132756, 0.00137974, 0.00133047, 0.
      ↪00123021, 0.00138492,
             0.12555005, 0.00117715, 0.03371139, 0.
      ↪11761563, 0.04584559,
             0.09084906, 0.04752858, 0.07862432])
```



7 Pose Adaptation

7.1 Inverse Kinematics

8 Results

9 Conclusion

10 Future Works

References

[1] 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

[2] 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.