## TABLE OF CONTENTS

Page

## LIST OF FIGURES

Page

## LIST OF ABREVIATIONS

| | |
|---|---|
| BiCG | Biconjugate Gradient |
| CGF | Computer Graphics Forum |
| DTFTM | Deformation Transfer for Triangle Meshes |
| EM | Energy Minimization |
| ÉTS | École de technologie supérieure |
| LBS | Linear Blend Skinning |
| NRR | Non rigid registration |
| PA | Procrustes Analaysis |
| VFX | Visual effects |

# INTRODUCTION

In the animation and visual effects industry, 3D characters are widely being used nowadays. They are often the most important animated objects in games, animations, and visual effects. In addition to modeling a character using 3D software such as Maya and Blender, a 3D character needs some pre-steps to get ready to be animated. These pre-steps are mostly complicated and time-consuming. This dissertation fits within a broader research project done at the École de technologie supérieure (ÉTS) Multimedia Laboratory through collaboration with an industrial partner, Autodesk, and involving the contribution of Quentin Avril, a postdoctoral fellow, Donya Ghafourzadeh and Srinivasan Ramachandran, PhD students, and Sarah Ribet, a professional's master student.

Animating a polygonal mesh by itself is seldom done directly and animators rely on high-level primitives such as skeletons. Thus, the character needs a skeleton consisting of a set of joints which connect the bones to each other. The quality of the motions resulting from the animation closely depends on the position of the joints in the skeleton as well as their orientation, thus this step is often done with high precision. The next step skins the character which sets the impact of each joint on each vertex, referred to as skin weights. In this sense, the transformations of the skeleton are linked to the transformation of the mesh through skin weights controlling which joints of the skeleton induce transformations to specific parts of the mesh. After applying an automatic binding method to generate the skin weights, those often need to be polished to make sure that the mesh will not have artifacts when animated. This can be done using the weight painting tools available in 3D modeling software. All these steps make the character preparation for animation very time consuming and complicated.

Transferring the whole animation setup of an already-done character to other 3D characters allows for preserving the artist work, reducing the artifacts, and therefore easing animation reuse and saving time. The animation setup refers to the skeleton including the rotation and

orientation of each joint, and skin weights. As the inputs, a ready to animate polygonal source mesh and a raw polygonal mesh as the target are required. Finding the corresponding location of each source joint on the target requires a vertex-to-point geometric correspondence between the source and target. Each joint on the source has a point cloud of its weighted vertices which represent the impact of that joint on the mesh. The point cloud of weighted vertices with respect to each joint is used to calculate the new location of the target joint with respect to its corresponding point cloud of weighted vertices on the target. The orientation of each joint is transferred from the source joint to the target joint directly, while the rotation alignment between the joint's weighted vertices point cloud on the source and its corresponding point cloud on the target leads to transferring the correct rotation of the joint, as the rotation and the orientation of the joints are very important features for animating a skeleton.

Various methods are published for either generating the skeleton and/or skin weights automatically, or transferring the skeletal structure and skin weights between characters. These approaches showed various types of limitations. Most of these methods such as the ones proposed by Miller *et al.* (2010) or Baran and Popović (2007) require a template or a large database of acceptable source meshes as input. Some are not capable to handle a wide range of skeletal structures, which limits the users' freedom. The "Animation Setup Transfer" proposes a general approach to overcome these limitations. This dissertation is mostly focused on the skeleton retargeting part of the global "Animation Setup Transfer" project and improving its results. Two different ways for transferring the skeleton are presented in this thesis, one by solving an energy minimization problem which is presented by Avril *et al.* (2016) and the other by taking advantage of Procrustes analysis which is presented as a part of this dissertation. In order to improve the skeleton transferring process, some techniques such as mirroring the joints and *spine alignment* are developed. Furthermore, to preserve the quality of the motions applied to the source character, a pose normalization is applied to the target meshes in order to make sure the joints in source and the target have the same orientation. These optimization and improve-

ment approaches are also presented as the the part of the "Animation Setup Transfer" by Avril *et al.* (2016).

Chap. 1 presents a literature review on the existing methods used for generating skeleton and weights, transferring the skeleton, and finding the geometric correspondence. In Chap. 2, the core of the "Animation Setup Transfer" approach by Avril *et al.* (2016) is described comprehensively. Chap. 3 presents two approaches for skeleton retargeting, one by solving an energy minimization problem and the other using Procrustes analysis. These methods are followed by the optimization approaches such as joint mirroring and *spine alignment* in order to improve the whole process. Several results using these optimization approaches and some comparisons to show the benefits of using the optimization approaches are presented in Chap. 4. Finally, a discussion around the approaches presented in this dissertation such as their limitations and potential suggestions to solve them is outlined in Chap. 5.

# CHAPTER 1

# LITERATURE REVIEW

In this chapter, an overview of methods related to character animation, skeleton and weight transferring as well as automatic methods to generate skeleton and skin weights is presented. Furthermore, related works to the methods of finding a geometric correspondence is discussed here, as this is a requirement for our further approaches.

## 1.1 Skeleton and Weight Creation

Some methods facilitate the process of generating a skeleton or creating the skin weights for a character already having a skeleton. Au *et al.* (2008) use Laplacian smoothing based on mesh contraction to extract a skeleton from a mesh. They iteratively remove the surface geometry in order to obtain a thin skeletal shape. Then a "surgery" is applied to the skeletal shape to remove the redundant connectivity and obtain a 1D structure. All of these steps are done using constrained Laplacian geometry smoothing and mesh simplification. In addition to producing a curve-skeleton, this method also generates the skeleton to vertex correspondence and a local "thickness". This method is limited to meshes with fine geometry (more than 5K of vertices) as it cannot generate fine skeletons form very coarse meshes. De Aguiar *et al.* (2008) use a deforming mesh sequence (mesh animation), with constant surface connectivity, to extract a kinematic bone hierarchy. However, it is not possible to locate a joint if there is no relative animation between adjacent body parts (e.g. in feet and hands due to insignificant relative animation in the feet or the fingers of the hands). Some methods like the ones by He *et al.* (2009) and Le and Deng (2014) require a set of example poses of a source mesh to calculate the skeleton and/or skin weights. He *et al.* (2009) take several poses of a given mesh as input. They use a harmonic function defined on the given example poses of the mesh and construct the skeleton-like Reeb graph from it. Then, the initial location of the joints are calculated by examining the changes in the mean curvatures. At the end, they refine the joint locations by means of solving a constrained optimization problem. Yet, this method cannot handle all

types of characters. Le and Deng (2014) compute the corresponding skeleton-based Linear Blend Skinning (LBS) model, including the skeletal structure and skin weights, to a set of example poses of a character mesh. The main limitation of this method is its low computational efficiency and the resulting artifacts from using the LBS model. Overall, all of these example based approaches share the same limitation of requiring various example poses of a source mesh as input, whose preparation is often a time consuming and complex process itself.

There are some methods which specifically generate skin weights. In the *Geodesic Voxel Binding* method by Dionne and de Lasa (2013), given a skeleton and a mesh, they are able to derive the skin weights by first voxelizing the input geometry, and then calculating the binding weights. Although this method works for production meshes that may contain non-manifold geometry or be non-watertight, applying its results may need post-process user interactions to modify the artifacts. The method of *Bone Heat (Heat Map Binding)* by Baran and Popović (2007) models the weight assignment for each bone as a heat diffusion system on the surface of the mesh. However in many cases, the results show artifacts which again require further user interaction to become more acceptable. On the other hand, some example-based methods to extract the skin weights are also available such as the methods by Le and Deng (2012), Li and Lu (2011), and Wang and Phillips (2002). Le and Deng (2012) aim to extract the skin weights given a set of example poses of a character. Requiring a set of example poses, is one of the limitations of this method. Li and Lu (2011) are able to automatically animate a model, but they require the skeleton and an animation (motion) of the skeleton, which is a limitation itself.

All in all, it can be summarized that the methods used for generating skeleton and skin weights are not able to provide the same level of quality as competent artists and generally require user interaction for a post-process to correct artifacts. Furthermore, most of them require a set of example poses or animations of a character as their inputs, which itself is a limitation.

## 1.2 Skeleton and Weight Retargeting

Some methods address retargeting a skeletal structure from one mesh to the other in order to ease the process of setting up a new character. The methods proposed by Poirier and Paquette (2009a) and He *et al.* (2009) retarget a skeleton from a source mesh to the target mesh using Reeb graphs to select the joint positions. Poirier and Paquette (2009a) adapt the given skeleton to the given character by matching topology graphs between them. He *et al.* (2009) present the skeleton transferring as an application for the cross parametrization. They compute a consistent harmonic 1-form of the source and target meshes, and then obtain the one-to-one correspondence between the isocurves of both. The skeleton can be transferred to the target taking advantage of this fact that each joint in associated with a unique isocurve. In the *skeleton sketching* method by Poirier and Paquette (2009b) the user interactively positions the joints which leads to facilitating the retargeting of the skeleton. The results obtained from these methods are limited to medial axis. Furthermore the skin weights on the target are not assigned automatically by the method, so an automatic binding method has to be used then. These methods rely on the *Bone Heat* method by Baran and Popović (2007) which then produces some artifacts and needs polishing.

Some methods such as the one presented by Baran and Popović (2007) get a generic skeleton as input and automatically adapt the skeleton within the given static character mesh. This method limits the artists from having a variety of skeletons as it only handles specific types of skeleton. Thus, it can not deal with hand or facial animation as it lacks the joints required for animating these parts. Moreover, the skeleton is often positioned incorrectly within the limb for two DOF joints such as knee and elbow as this method works with a reduced skeleton, in which all bone chains are merged.

Other methods such as *Skeleton driven animation transfer* by Chang *et al.* (2006) propose a system for transferring the animation. Given a well-edited character animation as the input, their proposed system is able to transfer the skeleton to another static character using consistent parametrization volume as the mapping between the space around two character meshes. The

main limitation of this method is that it only considers the parent link between joints, thus the problem of joint orientation is not handled (refer to Sec. 3.2 for further explanation about the importance of the joint orientation).

To recap, all of the methods used for retargeting a skeleton from a source character to a target character have different limitations. Some are only able to handle specific types of skeleton and limit the users to template skeletons or even input meshes (Baran and Popović 2007). Some of the methods ignore the joint orientation (Chang *et al.* 2006), and some constrain the joint positions (Poirier and Paquette (2009a,b), He *et al.* 2009). Besides, these methods do not allow for retargeting the skin weights but require computing them from scratch. To overcome these limitations, we propose two different ways for transferring a skeletal structure from a source mesh to one or many target meshes in Chap. 3.1. One using energy minimization which is published in "Animation setup transfer for 3D characters" by Avril *et al.* (2016) and the other, using *Procrustes analysis*.

There exist few methods which intend to transfer both skeleton and skin weights from one character mesh to another using either template characters or a large data base of rigged characters. Allen *et al.* (2003) propose a method to transfer the animation setup from one character to other scanned character meshes. They fit template meshes to 250 human body range scans using sparse 3D markers. Then, having the correspondence between all of these meshes using consistent parametrization, the skeleton pose and bone lengths are calculated using inverse kinematics. To transfer the skin weights, they use a skinning scheme based on per-vertex weights and the weights are transferred according to the correspondence. This method works for near-similar morphologies, but for morphologies with large differences, it does not result in decent joint positions. Also, discontinuities and animation artifacts may occur while transferring the skin weights between meshes with different polygon densities. The *Anatomy Transfer* method by Ali-Hamadi *et al.* (2013) uses an input template as the reference anatomical model and transfers the skeleton and skin weights to an arbitrary target character. The main limitation of this method is its reliance on template characters as it is not able to retarget user-defined skeleton and skin weights. The method proposed by Miller *et al.* (2010) (*Frankenrigs*) uses

a large database of fully-rigged characters. It finds, transfers, and blends weights from these characters to a target mesh and in this way it is able to overcome many limitation, but the main limitation lies in the creation of that large database itself as the system is as good as its database. The source character must contain a valid rigging to match one of the character meshes in the database, thus many characters with, for example, unusual limb size or shapes may not result in a desirable match from this database. Furthermore, the user is limited to use only the skeletal structures which are defined in the database. Generally, the methods used for transferring both skeleton and skin weights require a large database or they rely on template characters which limits the user's freedom. A summary of these methods and their limitations is shown in Table. 1.1. The approaches proposed in this dissertation and also the skinning weight transfer technique of "Animation Setup Transfer" overcome these limitations and go further by adding more features to this animation setup transfer.

Table 1.1 Available skeleton retargeting methods and their main limitations.

| **Methods** / **Limitations** | Poirier (2009a) | Baran (2007) | Chang (2006) | He (2009) | Allen (2013) | Ali-H (2013) | Miller (2010) | Avril (2016) | Thesis |
|---|---|---|---|---|---|---|---|---|---|
| Constraining the joint position | X | | | X | | | | | |
| Ignoring the out of the mesh joints | | | | | | | | X | |
| Ignoring the joint orientation | | | X | | | | | | |
| Limiting the user to specific skeletal structures | | X | | | | | X | | |
| Requiring template meshes | | | | | X | | | | |
| Requiring template mesh deformation | | | X | | | | | | |
| Requiring a database of fully rigged characters | | | | | | | X | | |
| Limited to near-similar morphologies and mesh densities | | | | | X | | | | |

### 1.3 Geometric Correspondence

Computing a meaningful geometric correspondence between a source and target mesh is often the first step to transfer the animation setup from the source to the target mesh. The more accurate and precise this geometric correspondence is, the more accurate the skeleton and skinning weight transferring will be. There are several methods which address this problem, each having their advantages and disadvantages. Our research group at the Multimedia laboratory of ÉTS intended to find the best available method according to the requirements of performing animation setup transfer. In this section, some key approaches in this matter are explained and compared to the one that is chosen for our approach. The reader can refer to recent surveys on this topic such as the ones by Van Kaick *et al.* (2011) or Orvalho *et al.* (2012).

The methods proposed by Lipman and Funkhouser (2009), Li *et al.* (2008), and Kim *et al.* (2010) are designed to find the a geometric correspondence only between isometric or near-isometric meshes in different pose. Isometric meshes are the ones which represent the same object in different pose, for example the same character with knees bent in different ways. According to our requirement for a geometric correspondence method being able to handle even non-isometric meshes and different morphologies, these approaches can not fulfil our expectation.

The methods of Sumner and Popović (2004), Aigerman *et al.* (2014), Ali-Hamadi *et al.* (2013), and Zell and Botsch (2013) are all able to handle non-isometric meshes. The goal of our work here was finding a decent method for computing the geometric correspondence and not inventing one. The methods used for comparisons are chosen according to the availability of their source code and the level of complication to implement them. The methods of Zell and Botsch (2013) and Sumner and Popović (2004) were found easier to implement, while providing the same quality of the results as the methods proposed by Aigerman *et al.* (2014) and Ali-Hamadi *et al.* (2013). However, between the *Deformation Transfer for Triangle Meshes (DTFTM)* method by Sumner and Popović (2004) and *Elastiface* by Zell and Botsch (2013), *Elastiface* showed better results according to our requirements of having bijection and smoothness in the

geometric correspondence. Fig. 1.1 shows a comparison between these two methods which is done by another student as a part of the "Animation Setup Transfer" project.

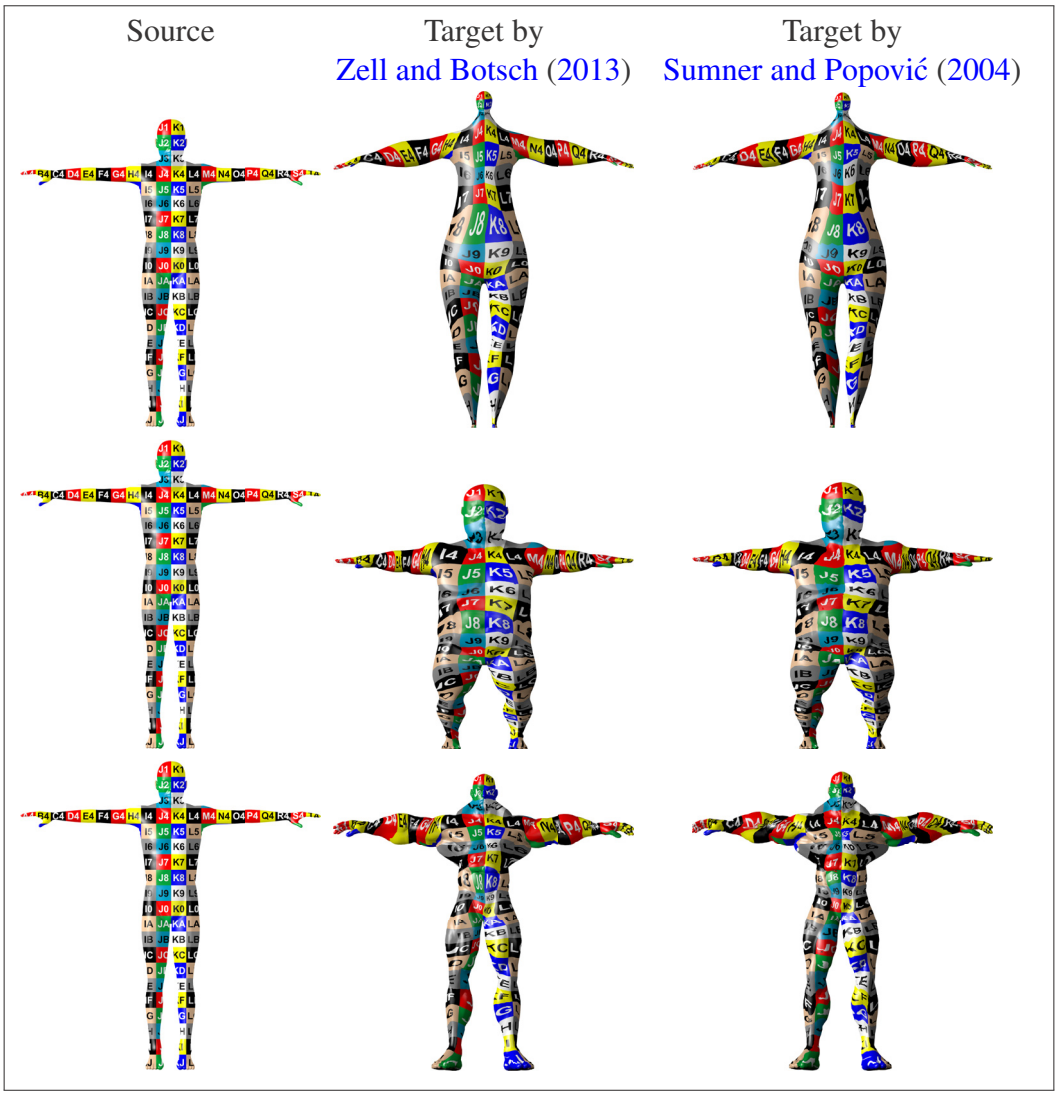



Figure 1.1 The comparison between the geometric correspondence methods of *Elastiface* and *DTFTM*. Although the results show similar qualities, the method of Sumner and Popović (2004) can produce unwanted distortions (see first and second rows). In the last row, *Elastiface* by Zell and Botsch (2013) produces significantly better correspondence for the character's arms. An orthogonal projective uv mapping is applied on the source and then transferred to the target based on the mapping.

# CHAPTER 2

## ANIMATION SETUP TRANSFER

This dissertation was done as part of a broader project called "Animation Setup Transfer", containing optimization approaches concentrating on transferring the skeleton from a source mesh to a target mesh. This chapter presents a brief but comprehensive explanation about the "Animation setup transfer for 3D character" paper by Avril *et al.* (2016). For more details the reader may refer to the full paper. Animation setup includes the skeleton (consisting of bones, joints connecting bones to each other, as well as orientation and rotation of each joint), and skin weights. The skin weights of a vertex refer to the impact that each joint has on the vertex. For each vertex there exists one weight value according to each joint. This value varies from zero (no impact) to one (maximum impact). A polygonal mesh is most often animated taking advantage of these high-level primitives.

The animation setup transfer from a ready-to-animate 3D character to one or many other 3D characters reduces the time spent on making a character ready to animate and it also preserves the artist work, thus leading to easy re-usability. Transferring the animation setup from a source character to one or many target meshes consists of three main steps. First, a geometric correspondence between the source mesh and the target mesh is found. Then the skeleton is retargeted from the source mesh to the target mesh. Finally, the skin weights are transferred, which leads to skinning the target mesh according to the source mesh's skin weights.

This method starts with a source mesh fully rigged (a hierarchy of joints and skin weights) and a target mesh as inputs. In the first step, the user provides the system with linked markers on both source and target meshes, emphasising similar semantic features (like the top of the fingers, the elbows, the knees, etc.). Fig. 2.1 shows an example of two characters and their set of linked markers. These markers act as the initial corresponding points from the source to the target. The number of markers by Avril *et al.* (2016) is 325, which was determined by testing different numbers and sets of markers in order to get a more accurate mapping.

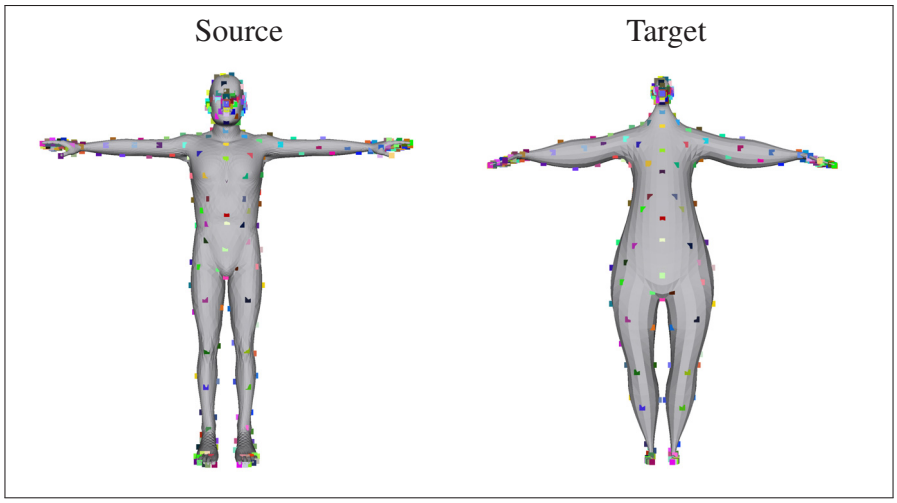



Figure 2.1 The Man character as the source and Curve character as the target with 325 linked markers to emphasize the semantic feature points of the meshes. The markers with the same colors are linked together.

The method of *Elastiface* by Zell and Botsch (2013), which is used by Avril *et al.* (2016), deforms the source mesh ($M_{\mathscr{S}}$) so that it matches the target mesh ($M_{\mathscr{T}}$). It then finds the dense vertex-to-point correspondence by finding the closest locations between the deformed $M_{\mathscr{S}}$ and $M_{\mathscr{T}}$. In the first step of deformation, the source is aligned so its position, orientation, and scale match the target's. Then, using a fairing technique, the source and target meshes get deformed to plain and featureless states, which share a very similar geometry. Fig. 2.2 shows an example of the source and target characters after the fairing stage.

Using the results from the fairing state, the non-rigid registration (NRR) of the source mesh is determined. Fig. 2.3 shows the Man character as the source after applying NRR, which deforms the source mesh to look like the smoothed target. Then, the NRR is applied again on the deformed source mesh so it matches the initial target mesh. The fairing and NRR steps involve solving energy minimization problems based on marker positions, vertex Laplacian, and Voronoi areas.

The correspondence is extracted from a per-vertex closest location between the deformed source and the initial target mesh. At the end, each vertex on the source is linked to a position

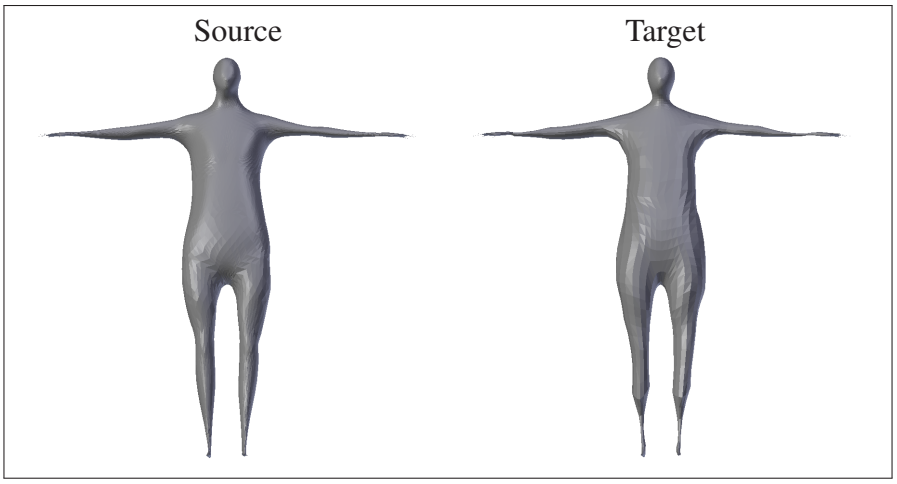



Figure 2.2 The Man character (left) and Curve character (right) after fairing. The source and target meshes are smoothed according to the markers and Laplacian values.

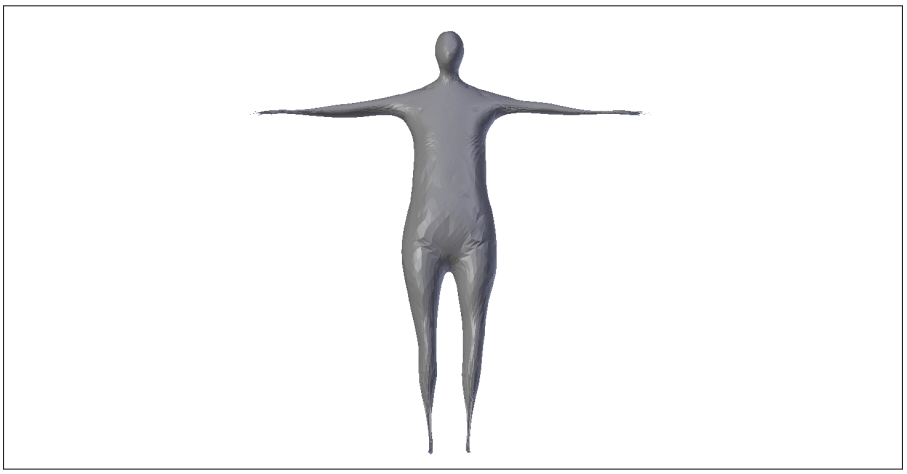


Figure 2.3 The source mesh after applying non-rigid registration. It deforms to match the faired target.

on the target by a barycentric coordinate and also in the same way, there exists a correspondence from each vertex on the target to a point on the source. Fig. 2.4 shows the mapping from a source to target mesh and vice versa using an orthogonal projective uv mapping. More details about the method of *Elastiface* by Zell and Botsch (2013) can be found in Appendix I.

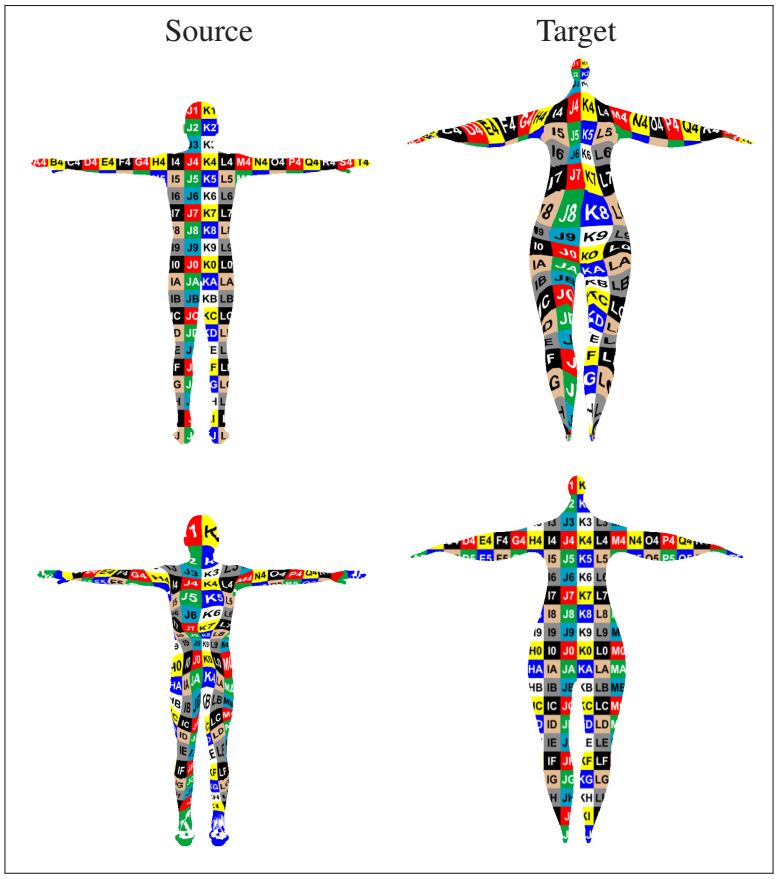



Figure 2.4 The geometric correspondence between the Man and the Curve character. The first row shows the mapping from source to target using an orthogonal projective uv mapping, which is applied on the source and then transferred to the target based on the geometric correspondence. The second row shows the reverse mapping from target to source.

After computing the geometric correspondence, the location of the source joints are determined within the target mesh. Fig. 2.5 shows a humanoid mesh with its skeleton. For each joint on the rigged source mesh, there is a point cloud of its weighted vertices which show the vertices influenced by the joint (only vertices with non-zero weight are considered). Fig. 2.6 shows this impact.

From the vertex-to-point geometric correspondence, the corresponding point cloud of weighted vertices for a specific joint can be found on the target. Extracting each joint position as a joint-vertices relationship and applying it on the target afterwards, yields to the new location of the

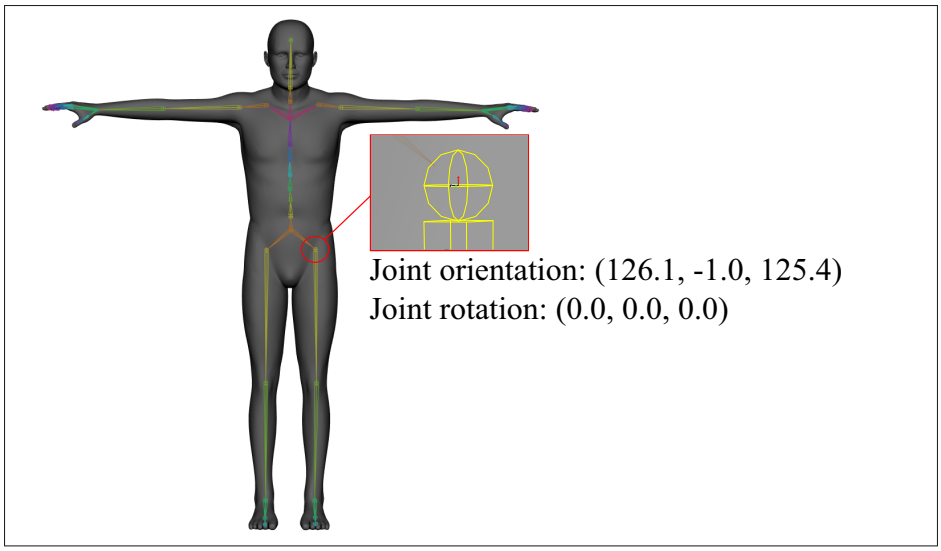



Figure 2.5 The Man character with its skeleton consisting of bones connected to each other by joints. The orientation and rotation of the left hip's joint is magnified in the figure. The rotation and orientation are in XYZ Euler angles local space with respect to its parent joint and in degrees.

target joint. In addition to each joint's location, the orientation of each target joint is copied from the source joint. At last, by finding the rotation alignment between source and target point clouds, the correct rotation of the target joint is also determined. Chap. 3 explains this step in detail as this is one the key contributions of this dissertation.

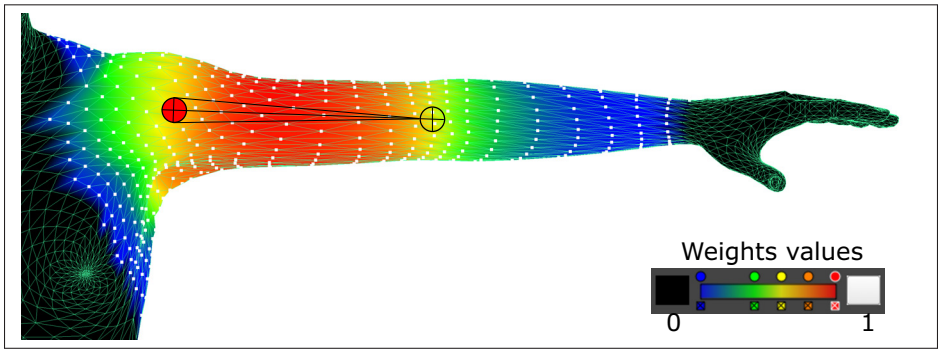



Figure 2.6 The point cloud of the vertices affected by the left shoulder joint (red joint) is highlighted in white. The colors show the impact of the joint on the vertices. Black shows zero weight and colors from blue to white correspond to weights of low to high value.

The last step in this process transfers the skin weights from the source to the target character. To transfer the skin weights related to each joint the reverse mapping from the target mesh to the source mesh is used. An interpolation from the source skin weights leads to new weights for the vertices on the target according to each joint. Fig. 2.7 shows an example of animated source and target characters after animation setup transfer.

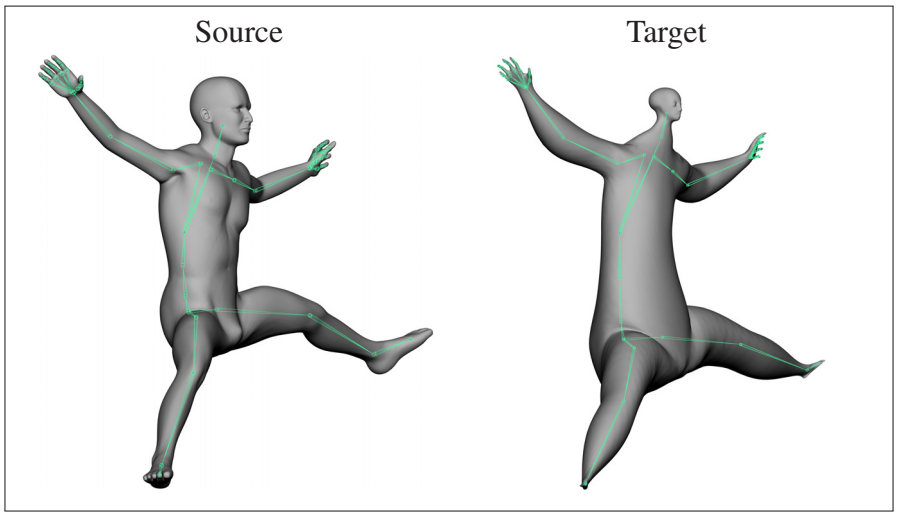



Figure 2.7 The animation setup including the skeleton and skin weights is transferred from the Man character to the Curve character. This figure shows a frame from the animation applied on the source and target.

# CHAPTER 3

# SKELETON TRANSFER

Transferring the source skeleton and determining its position within the target mesh is the key contribution of the author in the "Animation Setup Transfer" project. In this chapter, the skeleton retargeting approaches are presented in detail. Moreover, some optimizations and enhancement approaches are presented in order to improve the results of skeleton transfering. As it was presented in Chap. 2, the input required to transfer the animation setup is a source mesh fully rigged in the first place and an arbitrary target mesh. The source character contains a skeleton consisting of a hierarchy of joints connected to each other by bones, and also the skin weights, which represent the impact of each specific joint on the vertices. The relationship between the joints and the mesh provides a meaningful geometrical link, as in the skeleton-based animation the joints directly deform the mesh. As such, the geometrical link between weighted vertices and their corresponding joint should be kept as similar as possible between the source and target while the position, rotation and the orientation of the joints are transferred. In the following sections, the approaches to reach this goal are presented.

## 3.1 Joint Position

After computing a vertex-to-point geometric correspondence between the source and target characters, the joints from the source skeleton are transferred within the target mesh. During the transfer, the relationships and features found in the source character should be preserved. Positioning the joints within the target mesh while maintaining that relationship can be done using two different approaches. The first approach which is used in the method of Avril *et al.* (2016) is to find the relationship between the joints and vertices using the skin weights, and then reproduce this relationship within the target mesh. This is done by solving an energy minimization problem which helps keeping the relationships in the target mesh as close as possible to the ones in source mesh. The other approach to position the joints within the target character is by using the generalized Procrustes analysis. This approach takes advantage of

the geometric correspondence and uses the orientation, translation, and uniform scaling for aligning the point cloud of the joint's weighted vertices on the source to its corresponding point cloud on the target in order to position the joint directly. In the following sections these two approaches are described. Furthermore the comparison between both approaches can be found in Chap. 4.

### 3.1.1 Energy Minimization

Avril *et al.* (2016) present an energy minimization method in order to retarget the joint within the target mesh. This section explains this method in detail. According to Chap. 2, each joint affects a point cloud of vertices. Those vertices with non-zero weights are referred to as *weighted vertices*. We assume that using the skin weights, the position of each joint ($J_{\mathscr{S}}$) can be expressed as a linear combination of its weighted vertex positions ($v_{i\,\mathscr{S}}$). In this matter, a set of coefficients $c_i$ is required in order to build up this linear combination. From all possible answers for the set of coefficients $c_i$, here we are looking for the one which fulfills the following features specifically:

a. The difference between linear combination ($\sum_{i=1}^{m} c_i v_{i\,\mathscr{S}}$) and the actual joint position ($J_{\mathscr{S}}$) is minimized.

b. The values of the coefficients $c_i$ are as much as possible proportional to the corresponding skin weights.

c. The summation of the coefficients $c_i$ for the target are as much as possible equal to one.

According to these required features, a good set of coefficients for each source joint ($J_{\mathscr{S}}$) is determined by solving the following quadratic energy minimization function:

$$E(c_1,\ldots,c_m) = \omega_1 \left\| \left( \sum_{i=1}^{m} c_i v_{i\mathscr{S}} \right) - J_{\mathscr{S}} \right\|^2 \tag{3.1a}$$

$$+ \omega_2 \sum_{i=1}^{m} \left( c_i - \frac{w_{i\mathscr{S}}}{\sum_{j=1}^{m} w_{j\mathscr{S}}} \right)^2 \tag{3.1b}$$

$$+ \omega_3 \left( \left( \sum_{i=1}^{m} c_i \right) - 1 \right)^2 \tag{3.1c}$$

where $w_{i\mathscr{S}}$ refers to the skinning weight of vertex $v_{i\mathscr{S}}$ according to the joint $J_{\mathscr{S}}$. The parameters $\omega_1$, $\omega_2$, and $\omega_3$ are responsible to keep the influence of each feature balanced. Eq. 3.1a minimizes the difference between the linear combination of weighted vertices positions and the joint position. Solving only this term of the equation may lead to non-unique results while they all yield the same joint position. Moreover, Eq. 3.1a does not constrain the influence of the weighted vertices. This may lead to involving vertices with low skin weight values which are further from the joint. In this way, they are able to affect the final positioning of the joint in spite of their low skin weight value. Thus, it is necessary to increase the influence of those vertices with higher skinning weight. This is where Eq. 3.1b is used to keep the coefficients $c_i$ commensurate to the corresponding skinning weight $w_{i\mathscr{S}}$. Precisely solving Eq. 3.1b can yield to large errors for Eq. 3.1a. Hence, having $\omega_1 \gg \omega_2$ increases the influence of Eq. 3.1a to avoid such excessive impact. With the last term (Eq. 3.1c), a sum of the coefficient equal to one is preserved, which keeps the joint position in the convex hull of the weighted vertices. This helps to keep the joint inside the mesh. The desired parameters in the paper of Avril *et al.* (2016) which are also used in this dissertation, have been set to $\omega_1 = 1{,}000$, $\omega_2 = 1$, and $\omega_3 = 10$.

Eq. 3.1 is minimized by solving the linear system of partial derivatives equal to zero. This is solved using the Matlab linear solver. After solving the resulting linear system, the set of

obtained coefficients $c_i$ is used to determine the position of the joint $J_{\mathscr{T}}$ in the target mesh as:

$$J_{\mathscr{T}} = \sum_{i=1}^{m} c_i p_{i\mathscr{T}}, \tag{3.2}$$

where $p_{i\mathscr{T}}$ is the corresponding point on the target mesh to the vertex $v_{i\mathscr{S}}$ on the source mesh based on the computed geometric correspondence. Fig. 3.1 shows an example of the skeleton which is copied from the source character and then is transferred to the target character.

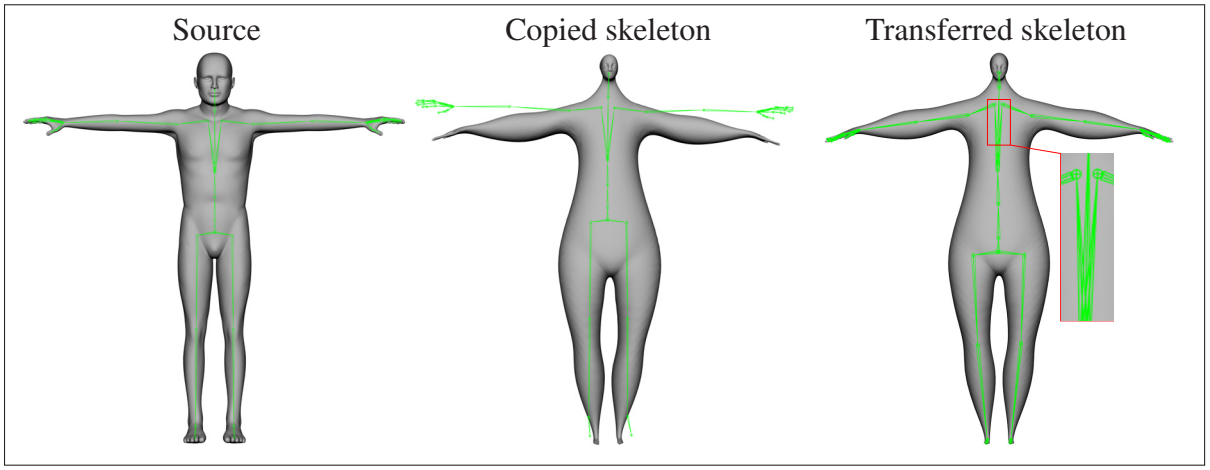



Figure 3.1 Source and target characters before and after skeleton retargeting using energy minimization. After copying the source's skeleton, it is correctly transferred within the target.

### 3.1.2 Procrustes Analysis

According to Gower and Dijksterhuis (2004), *Procrustes Analysis (PA)* is a multivariate technique which uses transformations (translation, rotation, reflection, and isotropic rescaling) to align a set of individual data (here, a set of weighted vertices) to another set of data as closely as possible. In this matter, *PA* is used to retarget each joint of the skeleton within the target character. This novel approach was implemented and developed as one of the main parts in this dissertation and is compared to the method used by Avril *et al.* (2016) (which was explained in Sec. 3.1.1) in Chap. 4. Taking advantage of the geometric correspondence computed in previ-

ous steps, the point cloud of weighted vertices for each joint in the source mesh corresponds to a point cloud in the target mesh. The transformations (here consisting of translation, rotation, and uniform scaling) that align the set of weighted vertices on the source to its corresponding point cloud on the target mesh will be equally good to transform the source joint to a meaningful location within the target mesh. Then, by applying these transformations to the initial source joint position, the new position of the joint within the target character is computed. To this end, the scaling ($S$), rotation ($R$) and then translation ($T$) transformations are applied to the source joint position ($J_{\mathscr{S}}$) to obtain $J_{\mathscr{T}}$. Fig. 3.2 shows an example of the whole skeleton retargeted within the target character using *PA*. Although the results are close to the energy minimization approach for the skeleton retargeting, the speed and simplicity of *PA* is notable. Further results and comparisons between energy minimization and *PA* can be found in Chap. 4.

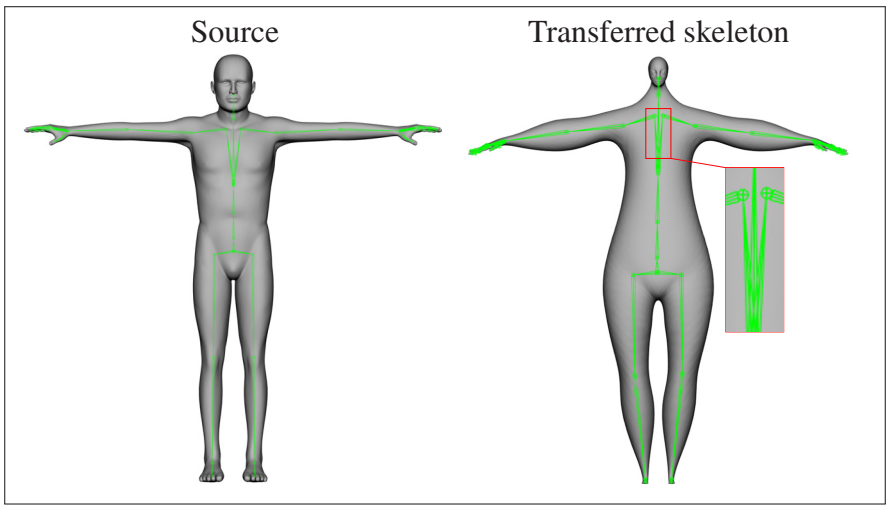



Figure 3.2 The Man character as the source and the Curve as the target character after skeleton retargeting using *PA*.

Considering all the vertices with non-zero weights can lead to involving even the vertices with an insignificant influence from the joint. Thus, a threshold is set for the skinning weight values so only the weighted vertices with a weight higher than the threshold are considered within the point cloud. The bigger the threshold is, the more limited the point cloud of the weighted

vertices will be. The threshold of 0.01 is set according to accuracy and quality of the resulting skeletons (refer to Chap. 4 for further comparisons between different thresholds).

### 3.1.3 Spine Alignment

Within the process of 3D character modeling, many modelers take advantage of available options in 3D modeling applications such as symmetrization and straight abdomen. Depending on the way modelers create characters with symmetry modes, some of these features can be used to improve our approaches for skeleton retargeting. For those meshes which have a straight abdomen, an approach is presented here to have a completely straight spine. This process is called *spine alignment* in the paper of Avril *et al.* (2016). A straight abdomen, and furthermore a straight spine, become useful when there are many abdominal moves in the animation. This alignment minimizes the artifacts and problems in the abdominal area during the animation process. As transferring the joint position, this also can be done either by solving an energy minimization problem, or by means of *PA*.

#### 3.1.3.1 Energy Minimization

The *spine alignment* using the energy minimization approach is done by modifying Eq. 3.1 and solving the linear system for those joints on the spine. Table. 3.1 highlights these modifications applied to Eq. 3.1. To this end, the user first provides the system with a symmetry plane (assume a lateral $yz$ plane). Then, Eq. 3.1 is changed to consider the weighted vertices related to all of the spine joints. Thus, $m$ changes to $m'$ showing the number of vertices with a non-zero weight for at least one spine joint (highlighted in red in Table 3.1).

In Eq. 3.1b, there were several weights $w_{i,\mathscr{S}}$, one for each vertex, and we were looping through each of those $m$ vertex weights for a single joint. We now deal with several joints, each having its own set of weights for each vertex. We thus need to sum up the weights per joint and per vertex. For each vertex, we first sum up its weights for all of the spine joints and store that value

Table 3.1 The top cell shows Eq. 3.1 which is used for calculating the coefficients in order to compute the joint position within the target. The bottom cell shows the modified equation which is used for *spine alignment*. The differences are highlighted in the same colour for each part of the equation.

| | |
|---|---|
| Joint positioning | $E(c_1,\dots,c_m) = \omega_1 \left\| \left( \sum_{i=1}^{m} c_i v_{i\mathscr{S}} \right) - J_{\mathscr{S}} \right\|^2$ (3.2a)<br>$+ \omega_2 \sum_{i=1}^{m} \left( c_i - \frac{w_{i\mathscr{S}}}{\sum_{j=1}^{m} w_{j\mathscr{S}}} \right)^2$ (3.2b)<br>$+ \omega_3 \left( \left( \sum_{i=1}^{m} c_i \right) - 1 \right)^2$ (3.2c) |
| Spine alignment | $E(c_1,\dots,c_{m'}) = \omega_1 \left\| \left( \sum_{i=1}^{m'} c_i v_{ix\mathscr{S}} \right) - J_{x\mathscr{S}} \right\|^2$<br>$+ \omega_2 \sum_{i=1}^{m'} \left( c_i - \frac{w_{i\,\mathrm{sum}\,\mathscr{S}}}{\sum_{j=1}^{m'} w_{j\,\mathrm{sum}\,\mathscr{S}}} \right)^2$<br>$+ \omega_3 \left( \left( \sum_{i=1}^{m'} c_i \right) - 1 \right)^2$ |

in $w_{j\,\mathrm{sum}\,\mathscr{S}}$. Thus, in Eq. 3.1b, $w_{i\mathscr{S}}$ is changed to $w_{i\,\mathrm{sum}\,\mathscr{S}}$ , and $w_{j\mathscr{S}}$ is changed to $w_{j\,\mathrm{sum}\,\mathscr{S}}$ (highlighted in blue in Table 3.1). Alg. 1 shows the way that new weights are determined.

**Algorithm 1:** Sum the weights for the weighted vertices of the spine joints.

```
/* Contains the weight of vertex i according to joint j        */
SpineWeightedVertices ← The union of weighted vertices for all of the spine joints.
for Each vertex i from SpineWeightedVertices do
    w_{i,sum,S} = 0
    for Each joint j on the spine do
        w_{i,sum,S} ← w_{i sum S} + w_{iS} w.r.t joint j
    end
end
```

Eq. 3.1a is also changed to only consider the $x$ values of the spine joints positions computed by solving the energy minimization problem (highlighted in green in Table 3.1). After computing

the new set of coefficients $c_i$, the $x$ coordinate of the target joints are determined as:

$$J_{x\mathscr{T}} = \sum_{i=1}^{m'} c_i p_{ix\mathscr{T}}. \tag{3.3}$$

Finally, the initially computed spine joint $x$ coordinates are updated with the new $J_{x\mathscr{T}}$. Fig. 3.3 shows an example of a character and its skeleton, before and after applying the *spine alignment* using the energy minimization.

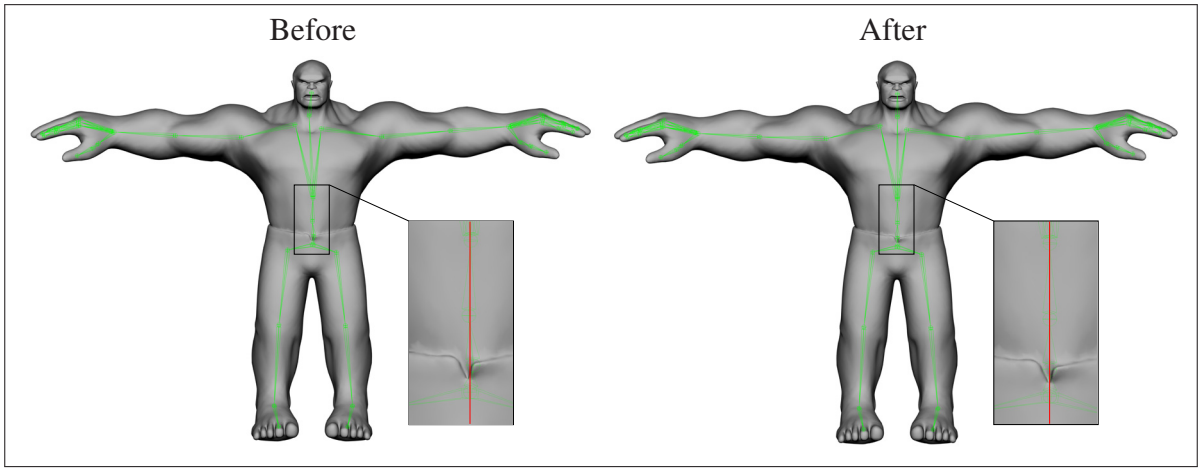



Figure 3.3 The Hulk character before and after applying the *spine alignment* using energy minimization. The spine joints closer to the same $yz$ plane after *spine alignment*.

#### 3.1.3.2 Procrustes Analysis

In order to align the spine joints using the *PA* approach, first the corresponding point cloud of all of the source spine joints' weighted vertices (with the same limits set as the weight value threshold in Sec. 3.1.2) is required. This is determined the same way as in the *spine alignment* using energy minimization. Then, by computing the rigid alignment between the point cloud of weighted vertices (for all spine joints) on the source and its corresponding point cloud on the target, the uniform scaling, rotation and translation are obtained. Afterward, the same as in the *PA* approach for retargeting the joints within the target mesh, the uniform scaling, rotation,

and translation transform the source spine joint positions one by one. Finally, the initial $x$ coordinate of each target spine joint position is updated by the new $x$ coordinates. As we have a single common transformation to compute the $x$ values for all of the spine joints, while they do not align perfectly, they tend to shift and drift much less than when using only the joint retargeting with PA. Fig. 3.4 shows the same character as Fig. 3.3 before and after *spine alignment* using *PA*. Further results obtained from applying *spine alignment* and comparison between using energy minimization and *PA* for spine alignment can be found in Chap. 4.

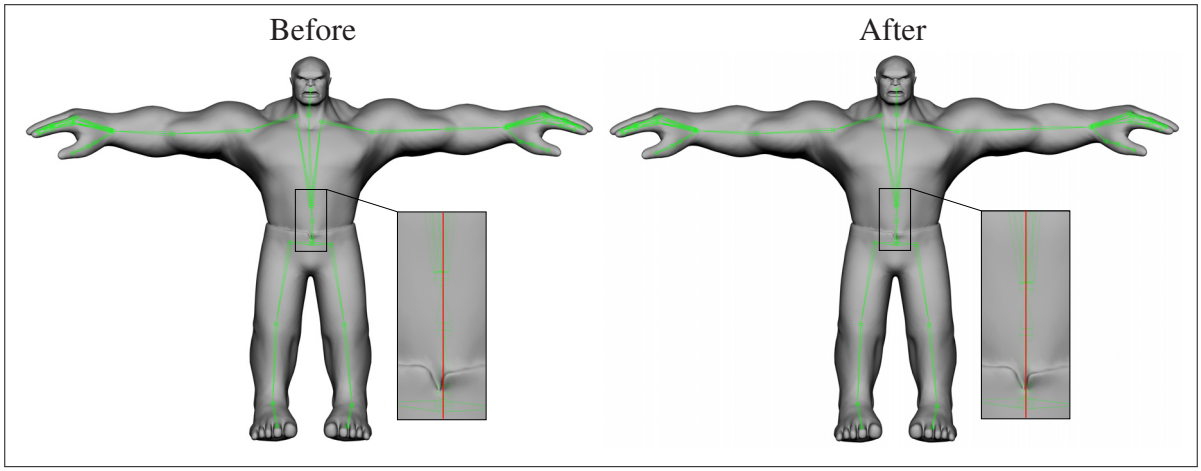



Figure 3.4 *Spine alignment* using *PA* for Hulk.

### 3.1.4 Mirroring

*Mirroring* is a post-process after optimizing the joint positions (using energy minimization, Sec. 3.1.1 or using *PA* Sec. 3.1.2), which leads to a completely symmetric skeleton. As it was shown in Sec. 3.1.1, the resulting target skeleton can be asymmetrical. For the characters which are symmetrical, an asymmetrical skeleton can be problematic. Thus, to fix the skeleton's asymmetry, the *mirroring* approach is applied to the target skeleton.

In the first place, the user provides the system with a symmetry plane. Assuming a sagittal (lateral) $yz$ plane as the symmetry plane, all the joints (excluding the joints on the spine) are symmetrized according to it. After positioning the joints within the target mesh, for each joint,

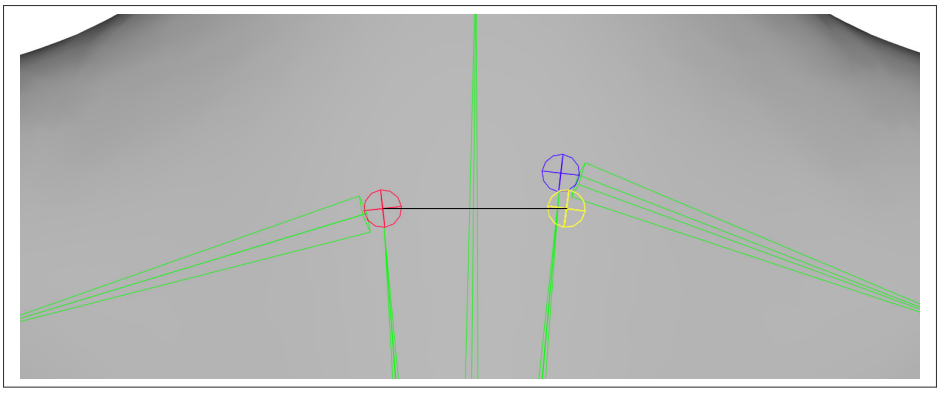


Figure 3.5 The shoulder joints of the Curve character from front view, after skeleton retargeting. $J_{Linitial}$ is shown in blue, $J_{Rinitial}$ in red, and $J_M$ in yellow. The final position is computed by averaging the position of $J_M$ and $J_{Linitial}$.

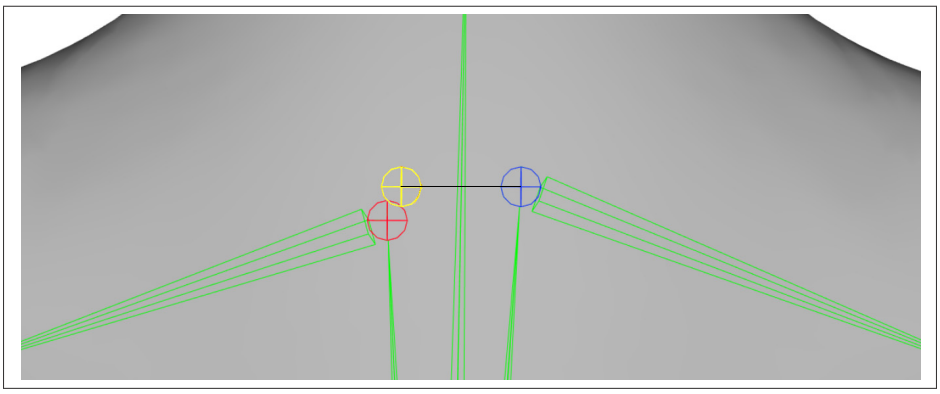


Figure 3.6 The shoulder joints of the Curve character from front view, after skeleton retargeting. $J_{Linitial}$ is shown in blue, $J_{Rinitial}$ in red, and $J_{M'}$ in yellow. The final position is computed by averaging the position of $J_{M'}$ and $J_{Rinitial}$.

the corresponding joint on the other side of the symmetry plane is determined. Assuming a joint on the right side of the symmetry plane ($J_{Rinitial}$ shown in Fig. 3.5 in red), the corresponding joint on the left side is $J_{Linitial}$ (shown in blue). Computing the mirror position of $J_{Rinitial}$ to the left side ($J_M$), the final position of the $J_L$ is computed by averaging the position of $J_M$ and $J_{Linitial}$. This process is applied on all the joints on the right and the left side of the symmetry plane considering the initial positions of the joints from skeleton retargeting process. In this regard, as we mirrored the right shoulder joint to the left side, we also mirror the left shoulder

joint to the right side and compute the average. Thus, in order to mirror the left shoulder joint ($J_{Linitial}$ shown in Fig. 3.6 in blue) this time, the same process is applied to it considering the original positions of the joints. It means that considering $J_{Linitial}$ its corresponding joint on the right side of the symmetry plane is determined which is $J_{Rinitial}$ (shown in red in Fig. 3.6). Then the mirror position of $J_{Linitial}$ is computed as $J_{M'}$ (shown in yellow in Fig. 3.6). Afterward, the final position of $J_R$ is computed as the average of $J_{Rinitial}$ and $J_{M'}$. The position of all of the joints are updated after all of them are gone through the mirroring process once. Fig. 3.7 shows an example of a character skeleton, before and after applying the mirroring process. Pay attention to how joints share the same $y$ and $z$ values with a mirrored $x$ value according to the symmetry plane.

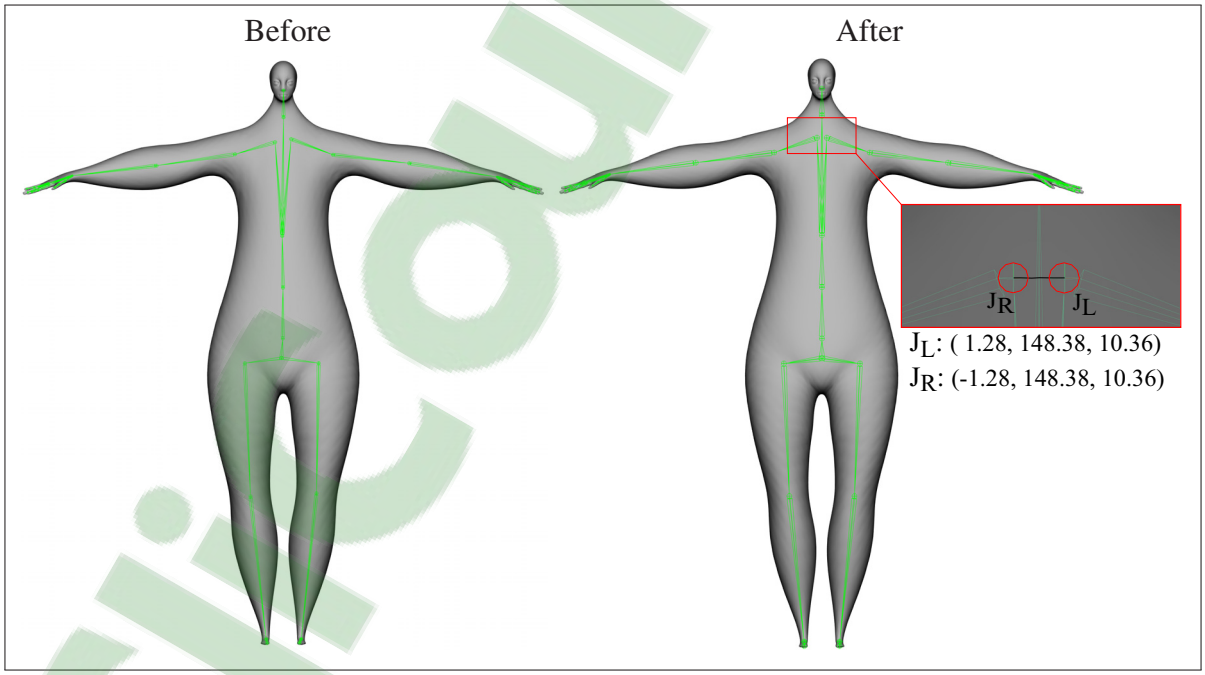



Figure 3.7 The Curve character before and after the skeleton mirroring. The shoulder joints are perfectly symmetric after applying the mirroring approach. The symmetry plane is set as $x = 0$ which is according to the root joint.

## 3.2 Joint Orientation and Rotation

The animation setup, as it was discussed in Chap. 2, also accounts for the relationship between the mesh vertex positions, skin weights, joint positions in the skeletons, and the joint rotations and orientations. One of the most important limitations of some of the skeleton transferring methods such as the method of Chang *et al.* (2006), is that it does not handle the joint orientation and rotation. Computing the joint position within the target mesh only leads the skeleton to be correctly placed within the target mesh, but for a right animation, a correct joint orientation and rotation is also required. Correctly transferring the joint orientation and rotation from the source to the target skeleton eases the motion remapping, which ensures that animations from a source character lead to similar poses in the target character, and also preserves the skeletal structure completely. Generally, *rotation* refers to a transformation attribute, which changes while the character is being animated. On the other hand, the *orientation* is a joint attribute specified for each joint, which sets each joint's local axes and is always fixed during the animation process. Fig. 3.8 shows the rotation and orientation properties in Maya.

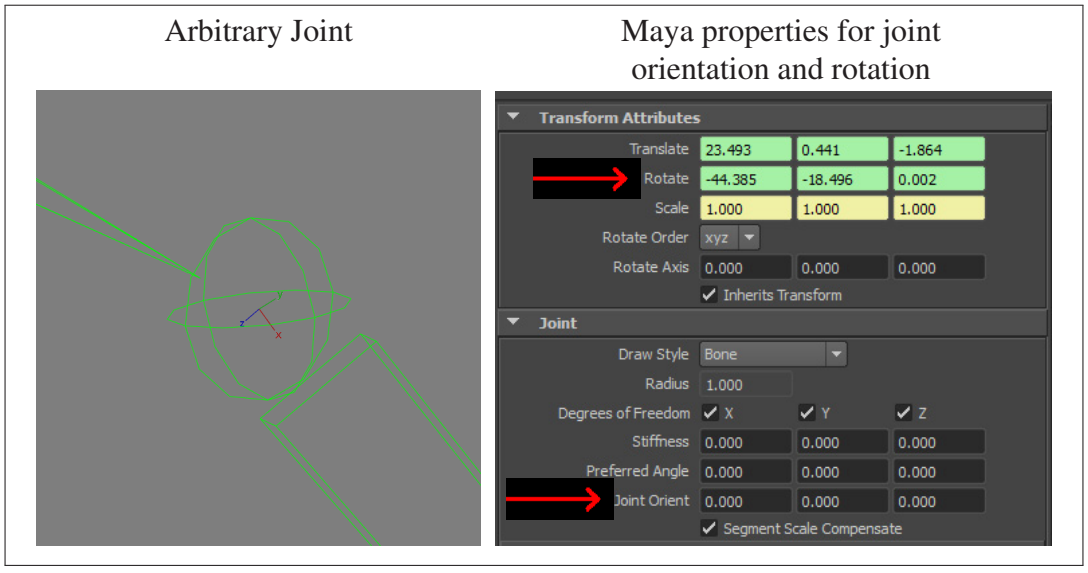



Figure 3.8 An arbitrary joint shown on the left and its rotation and orientation properties in Maya on the right.

The orientation and rotation both can be local or global. The global rotation refers to the rotation along the original global coordinates, which is set for each joint independent from the parent joint's orientation and rotation. In contrast, local rotation refers to the rotation of the joint local axes with respect to its parent's joint local axes. The same applies to the orientation. Fig. 3.9 shows the difference between the global and local orientation values in a very simple 3D mesh.

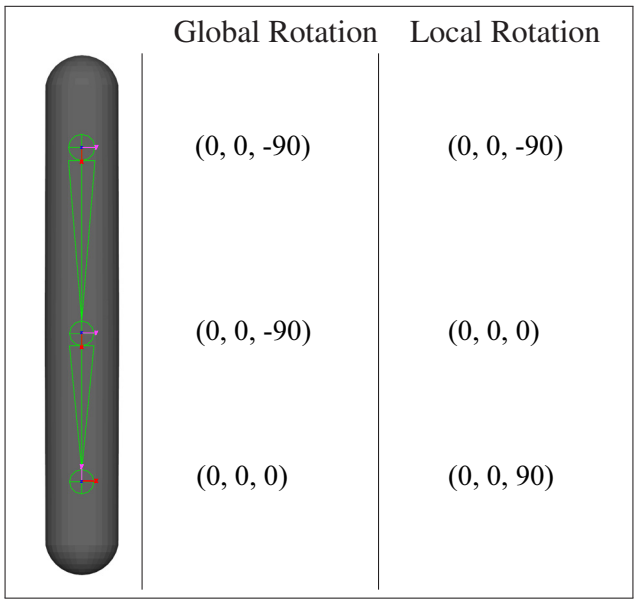



Figure 3.9 A simple 3D mesh and its three joints showing both global and local orientations. The local orientation of each joint is along its parent's local axes while the global orientation is along the world global coordinates. Here, the top joint is the root joint, so its local orientation is along the world global axis, which leads it to be equal to its global orientation.

To correctly transfer the joint's orientation and rotation, the already calculated vertex-to-point geometric correspondence between the source and target is used. Relying on the point cloud of the source weighted vertices $\{v_{i\mathscr{S}}|w_{i\mathscr{S}} > 0\}$ and their corresponding locations on the target $\{p_{i,\mathscr{T}}\}$, each joint is processed at a time. Beginning from the root joint, the local orientation of joint $J_{l\mathscr{S}}$ is copied directly to $J_{l\mathscr{T}}$. Then, the rotation required for aligning the point cloud of the source joint weighted vertices ($\{v_{i\mathscr{S}}|w_{i\mathscr{S}} > 0\}$) onto the point cloud of $\{p_{i,\mathscr{T}}\}$ is

determined using the best-matching similarity rotation by Umeyama (1991) which leads to a rotation in global space. The computed rotation in global space is then converted to a Δrotation in the local space of the joint $J_{l\mathscr{T}}$. This process is done recursively visiting each joint, and rotating the points $\{p_{i,\mathscr{T}}\}$ based on the growing inverse Δrotation of the kinematic chain starting from the parent joint and going up to the root joint to calculate the new Δrotation. Alg. 2 shows the detail of the this process, which starts by calling the following function for the root joint:

$$\textbf{TransferJoint}(J_{\text{root}\,\mathscr{S}}, J_{\text{root}\,\mathscr{T}}, \text{identityMatrix})$$

**Algorithm 2:** Transferring the joints orientations and computing their rotations.

```
1  TransferJoint (J_lS, J_lT, parentTransf) ;
2  begin
3      J_lT.orient = J_lS.orient ;
4      {p_iT} = GeometricCorresp ({v_iS | w(v_iS, J_lS) > 0}) ;
       /* Considering the rotations from parent joints                */
5      {p'_iT} = Rotate ({p_iT}, parentTransf) ;
6      globalRot = BestRot ({v_iS | w(v_iS, J_lS) > 0}, {p'_iT}) ;
7      J_lT.Δrot = J_lS.globalToLocalMatrix * globalRot ;
       /* Move to the next joints for both the source and target      */
8      foreach child_S ∈ J_lS.children ; child_T ∈ J_lT.children do
9          inverseRot = parentTransf * (−J_lT.Δrot) ;
10         TransferJoint (child_S, child_T, inverseRot) ;
11     end
12 end
```

Fig. 3.10 shows an example with and without handling the joint orientation and rotation correctly, which can affect directly the resulting animation transferred from the source to the target.

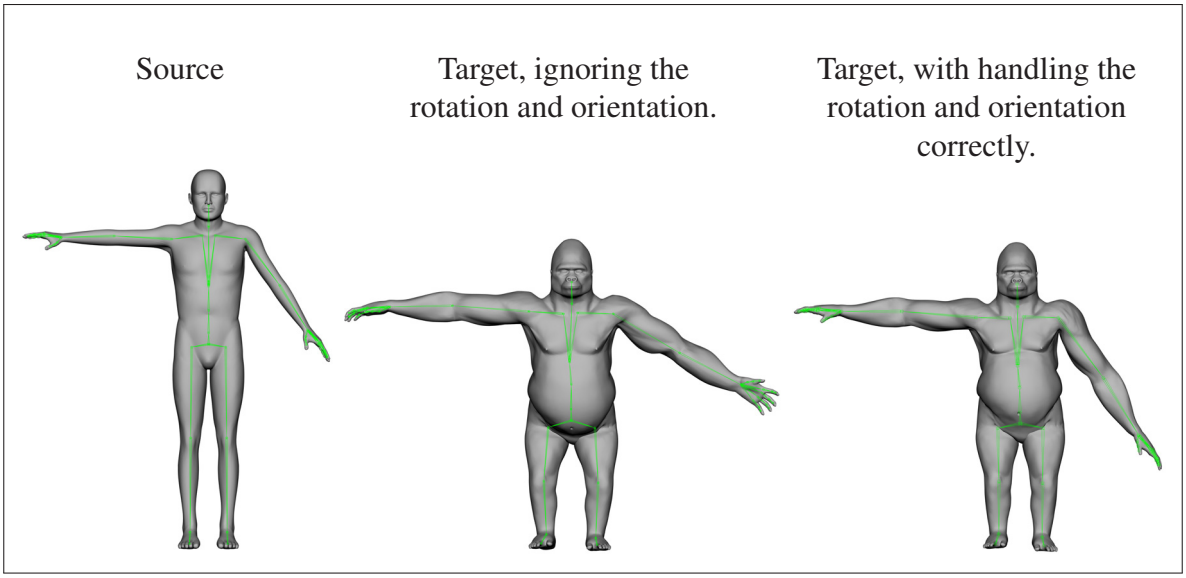
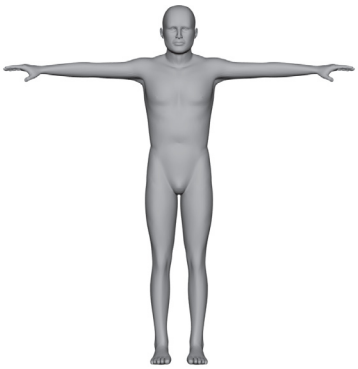



Figure 3.10 After transferring the animation setup from the Man to the Gorilla character, a transformation is applied to the Man's arm (source). This transformation is also applied to the Gorilla characters (target) within the forward kinematic animation. The first column shows the results when the orientation and rotation are not handled correctly and the second shows the same character and skeleton with correctly transferred joint orientation and rotation using our approach. When the orientation and rotation of the target mesh is handled correctly, the behaviour of the target should be the same as the source character within the forward and inverse kinematics.

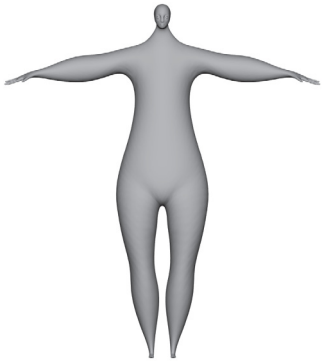
## 3.3 Pose Normalization

After retargeting the skeleton within the target mesh, as it was discussed in Sec. 3.1 and 3.2, the orientation of the joints are copied from the source joints orientations. While all the characters that were used in this dissertation were more or less in similar T-poses, there existed some differences in limb orientation. Fig. 3.11 shows some of these differences.

Thus, since the target character may not be exactly in the same neutral pose as the source character, $\Delta$rotation computed in Sec. 3.2 for each joint are used to put the target character in the same pose as the source. This operation is referred to as *pose normalization*. This is done by modifying the joint rotation using the $\Delta$rotation. Visiting each joint in the target skeleton, it is rotated by the inverted $\Delta$rotation, so that the actual rotation of each joint matched the rotation of the same joint in the source skeleton. This process considerably improves the



a) Man character: arms horizontal, legs vertical, feet straight ahead.



b) Curve charactr: Arms toward A-pose, legs inward.

c) Muscle character: Arms horizontal, legs outward, feet outward.

Figure 3.11 Differences in poses for different characters. Some characters have perfectly horizontal arms (a), (c) while others are pointing slightly downward (b). Legs are sometimes perfectly vertical (a), but the legs for some characters are pointing inward (b) or outward (c), and the feet of some characters are pointing straight in front (a), whereas for the others they are rolled outward (c).

character's animation process since it helps the characters to start from and end up with the same position as source character before and after applying the same animation. This leads to easing the animation reuse. Fig. 3.12 shows an example of a target character before and after *pose normalization*. Note the way that all the limbs of the target character match the reference pose of the source character. More examples of the results obtained by this approach, will be shown in Chap. 4.

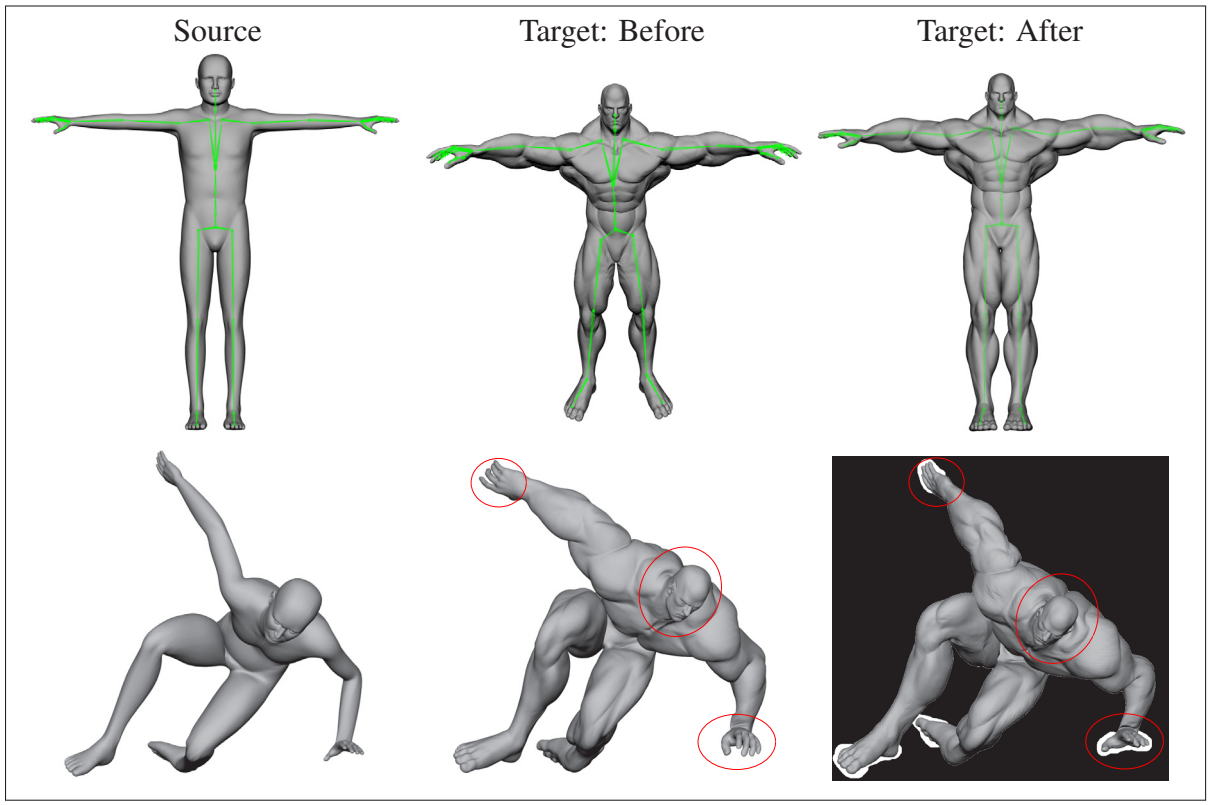



Figure 3.12 The Man character as the source and the Muscle character as the target, before and after applying the *pose normalization*. Second rows the the source and target within an animation, which is applied to both. Note the differences that *pose normalization* makes in the limb orientation within the animation.

# CHAPTER 4

## RESULTS AND COMPARISONS

This chapter presents further results which were obtained using the approaches proposed approaches in this dissertation. The mathematical computations were mostly done using Matlab. Then, the results are extracted in the form of text files. These text files are imported by the plugins written for Maya using python. The final results are compared and tested, generally by applying animations to the target characters to validate the skeleton and the skin weights.

Table 4.1 Characteristics of the twelve characters used for validations.

| Character Name | Number of Vertces | Number of Triangular Faces | Source |
|---|---|---|---|
| Bulk | 20,087 | 40,087 | Autodesk Character Generator |
| Curve | 4,137 | 8,270 | Designed by Joël Morency |
| Dwarf | 15,950 | 31,912 | Designed by Joël Morency |
| Elf | 1,340 | 2,676 | tf3dm.com |
| Gorilla | 7,632 | 15,260 | The TOSCA dataset by Bronstein *et al.* (2006) |
| Gorn | 20,371 | 40,738 | Autodesk Character Generator |
| Hulk | 3,869 | 7,734 | tf3dm.com |
| Kayan | 16,233 | 32,466 | Designed by Joël Morency |
| Macrocephalic | 23,571 | 47,138 | Designed by Joël Morency |
| Man | 10,196 | 20,388 | Autodesk Character Generator |
| Muscle | 12,204 | 24,404 | tf3dm.com |
| Sumo | 8,361 | 16,718 | Designed by Joël Morency |

Once the whole skeleton and the skin weights were transferred from the source character to the target character, the character is ready for animation. Both inverse and and forward kinematics can be used in this matter. Table. 4.1 shows the information for the characters used to validate the proposed approaches. Different characters with different number of vertices and faces, different resolutions, and different topologies can be handled by the proposed approaches.

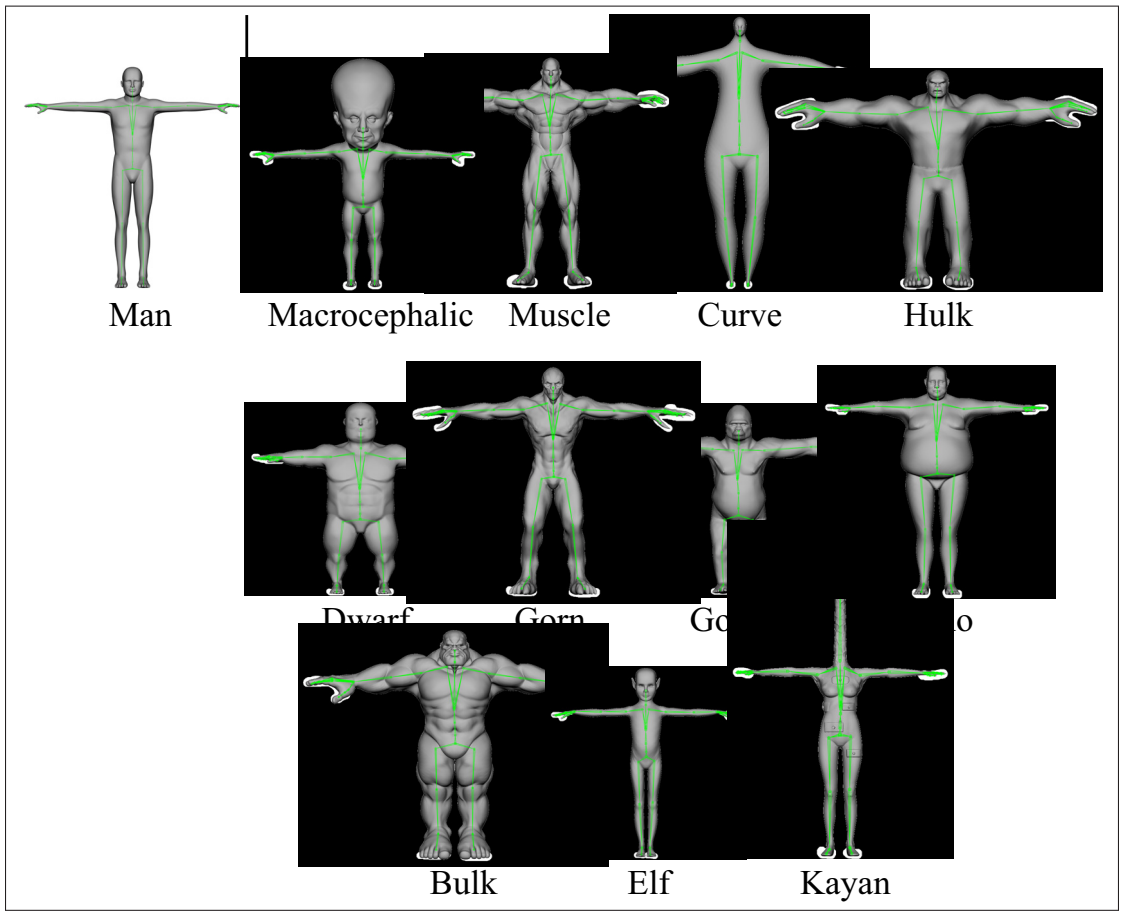



Figure 4.1 All of the characters used for validating the approaches. The left hand side shows the ready-to-animate Man character as the source and right hand side shows all of the other eleven target characters with their transferred skeletons and weights using the energy minimization approach. Target meshes notably have significantly different shapes, mesh topologies, numbers of vertices and faces, as well as lengths and diameters of limbs.

Fig. 4.1 shows all of these characters. The man character was used as the source. Its skeleton contains 52 joints and 51 bones which connect them to each other. These characters have

morphological and topological differences such as the different size of the limbs in different characters. In spite of these differences, the skeleton was correctly scaled to fit into each of the target characters, containing the the head, arms and fingers down to the spine, hips, legs and toes. The meshes varied from low resolution (2,676 triangular polygons for the Elf) to high resolution (50,872 triangular polygons for the Gorilla). These characters were chosen because of their variety of mesh topology and morphological features. For instance the big forehead of the Macrocephalic, tiny head and big curves of the Curve, big belly of Sumo, or long neck of the Kayan verify the generality of this approach. According to this fact that hands and fingers play a key role in most animations and artists mostly use a large number of joints to have a better control over those areas, being able to handle the big hands of the Hulk and Bulk as well as the Curve character's tiny hands covers a wide range of possibilities. Moreover, correctly transferring the skeleton to the Gorilla, which has shorter legs than its arms in comparison with humanoid characters, adds another dimension to the generality of the proposed approaches.

Furthermore, the skeleton and weights can be transferred between characters even with mismatched mesh topologies without any manipulation to their meshes. This brings more freedom to the artists in the industry as they can transfer the skeleton and weights among proxy characters, game characters built from a small number of polygons, and visual effects characters built from a big number of polygons . Fig. 4.2 shows all of the characters after applying the whole animation setup transfer using the energy minimization approach. The animation of the muscular limbs of the Muscle and Bulk characters (shown in Fig. 4.2) also validates the accuracy of the weight transfer. This generality is not limited only to energy minimization approaches. Fig. 4.3 shows the same characters after correct skeleton retargeting using the *PA*.

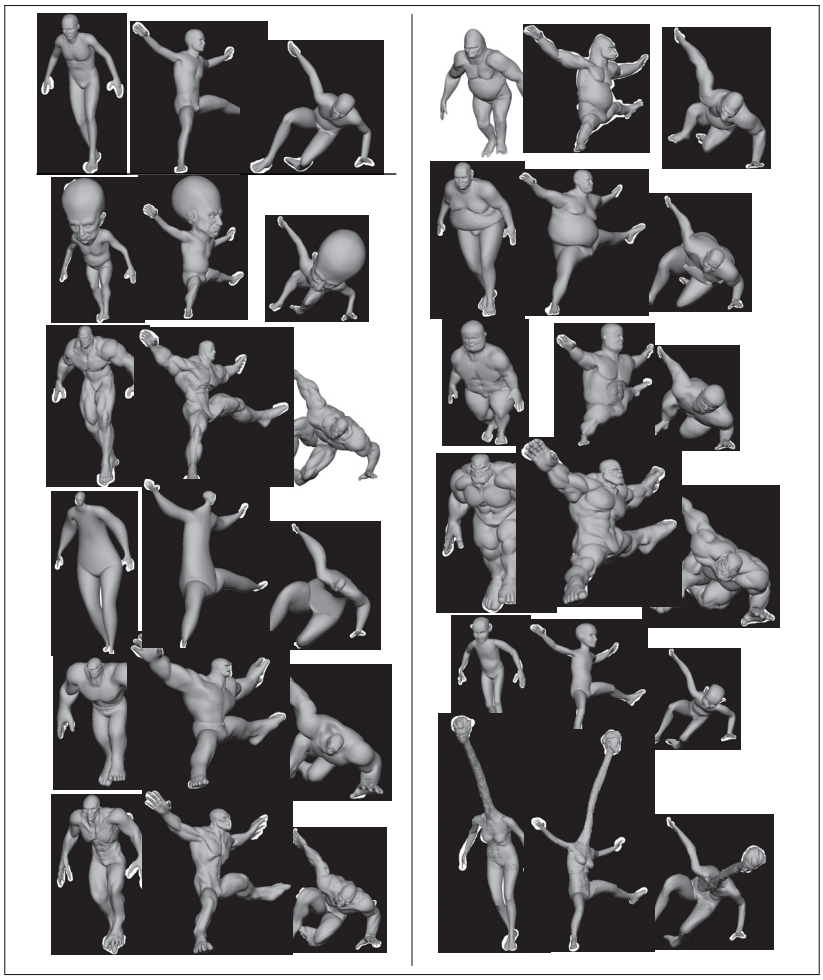


Figure 4.2 All of the twelve characters, after animation setup transfer from the Man to the other characters, using energy minimization. The approach is validated through a dance animation using inverse kinematics. Three different frames of this animation are shown in this figure for each character.

Moreover, the approaches are not limited to a specific type of source characters. Fig. 4.4 shows some examples of different characters as the source.

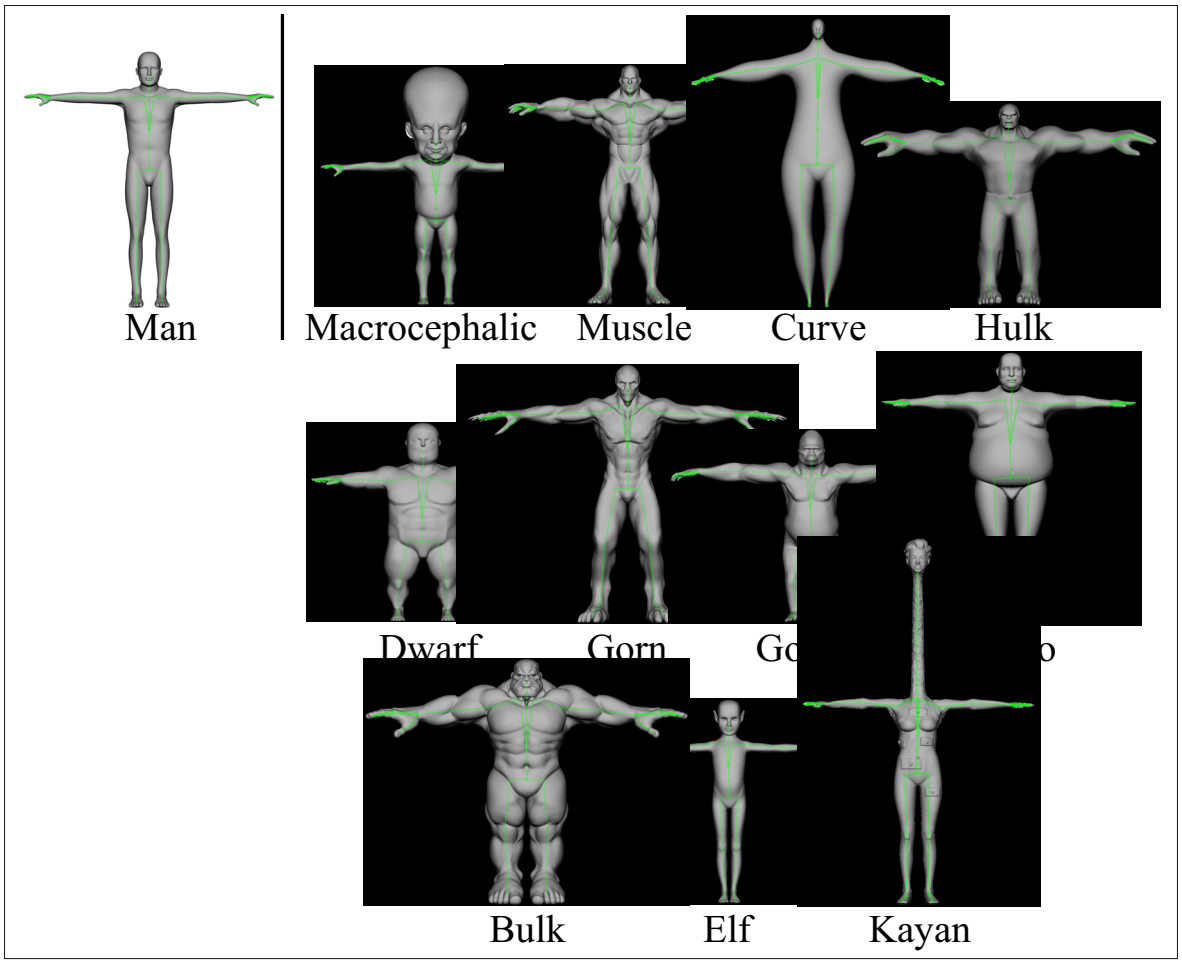
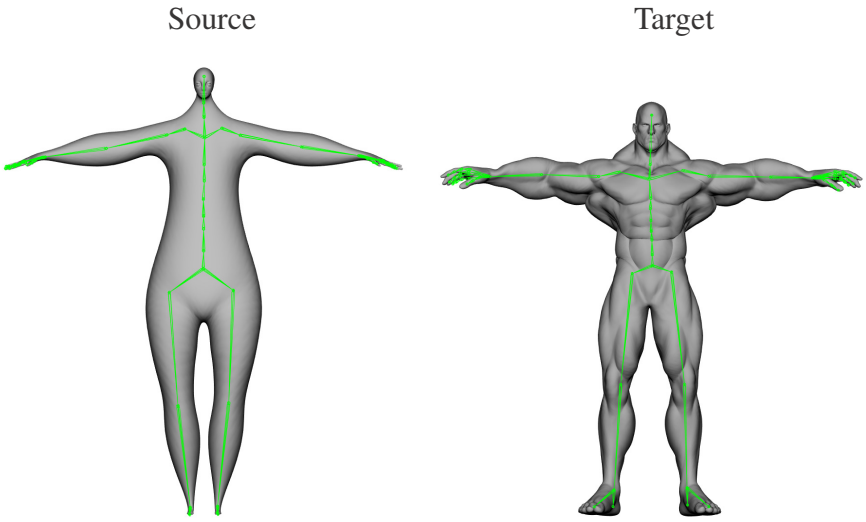
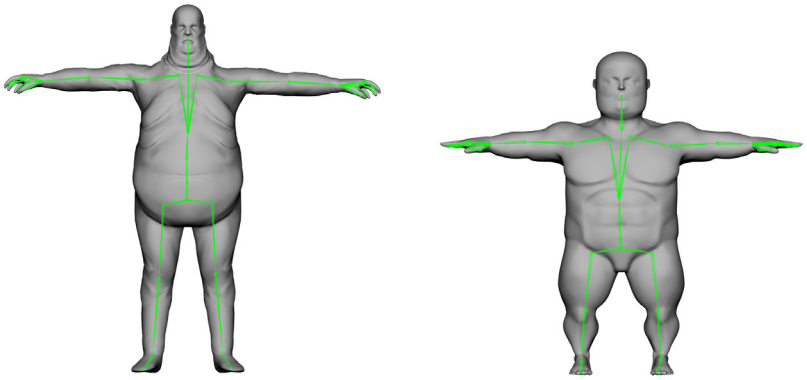



Figure 4.3 The Man character as the source and all of the other eleven characters with their skeleton correctly transferred using *PA*.




a) Retargeting from the Curve to the Muscle.



b) Retargeting from the Bloat to the Dwarf.

Figure 4.4 Example of retargeting with other source characters.

## 4.1 Artist Work Preservation

One of the greatest advantages of the proposed approaches is the ability of transferring the artist work from one mesh to others. For instance, to simulate the anatomical movement of the knee correctly, the knee joints are placed close to the mesh surface by most of the artists. In contrast, most of the rigging tools typically use the projected centering technique to set the joint on the medial axis. Thus, additional, mostly time-consuming, editing operations would be required to create a skeleton from scratch for each target character. The skeleton transfer approaches which were proposed in this dissertation more faithfully reproduce the artist work

(Fig. 4.5). Furthermore, the same applies to weight editing where every editing on the source skin weights directly impacts on all the target meshes. The weight editing preservation was tested by other students within the “Animation Setup Transfer” project.

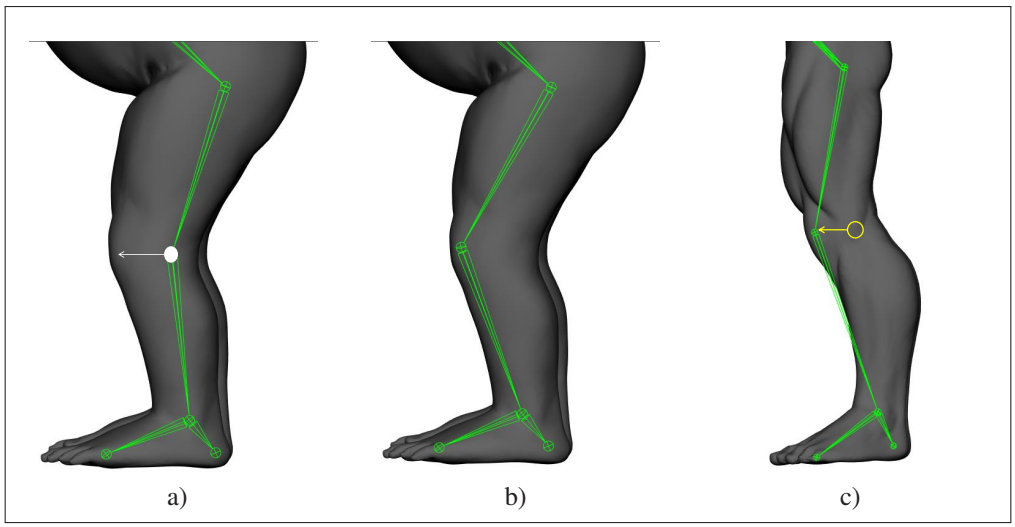



Figure 4.5 Our approach is not limited to specific joint placement (e.g. in the medial axis of the limbs). Two different joint positions – (a), (b) – on the Gorilla character are correctly retargeted to the corresponding positions on the Muscle character (c).

Moreover, our approaches are not limited to a specific form of skeletons. Fig. 4.6, 4.7, and 4.8 show different variation of skeletons which were correctly transferred from the Man character to the Macrocephalic. Thus, artists have more freedom to design skeletons with any number of joints and/or a different hierarchy according to their animation purpose.

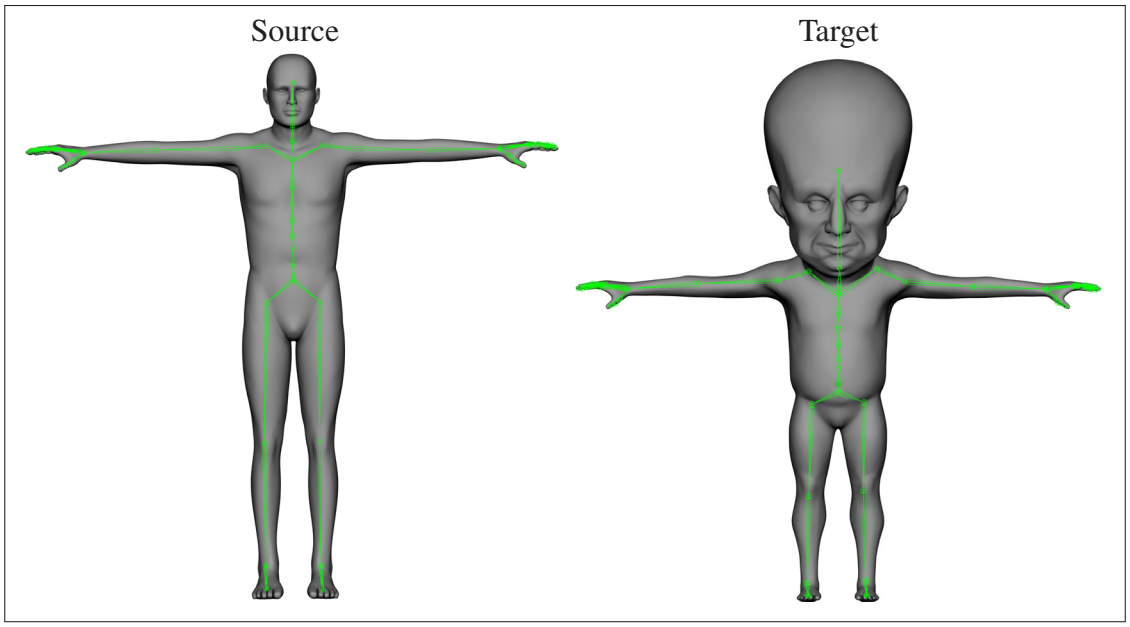



Figure 4.6 The Man and the Macrocephalic characters with a skeleton consisting of 67 joints.

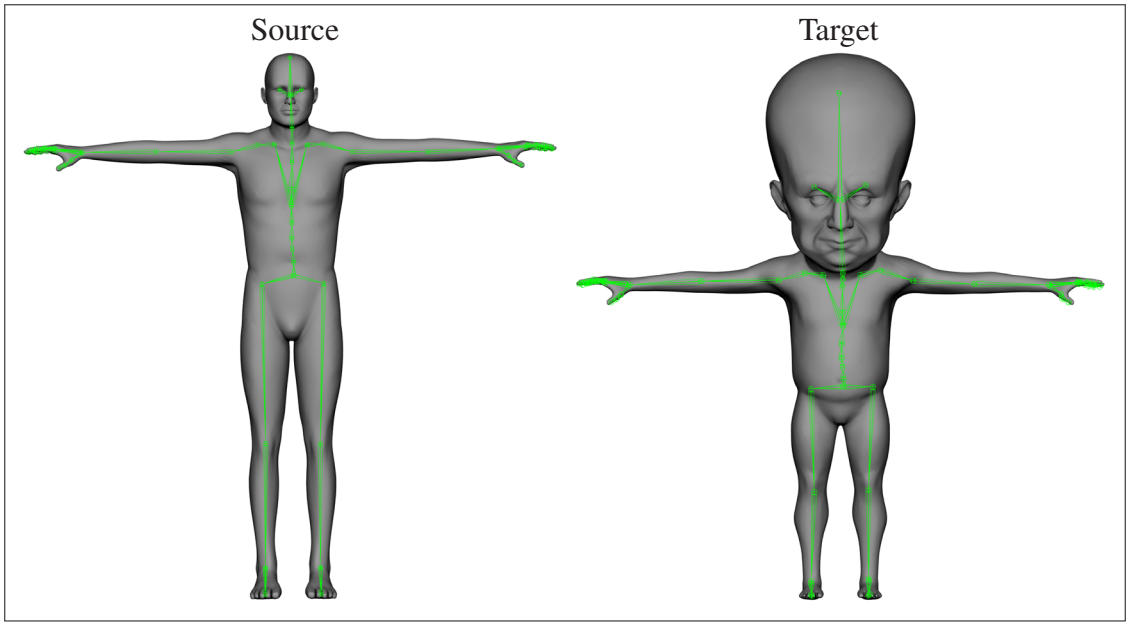



Figure 4.7 The Man and the Macrocephalic characters with a skeleton consisting of 72 joints.

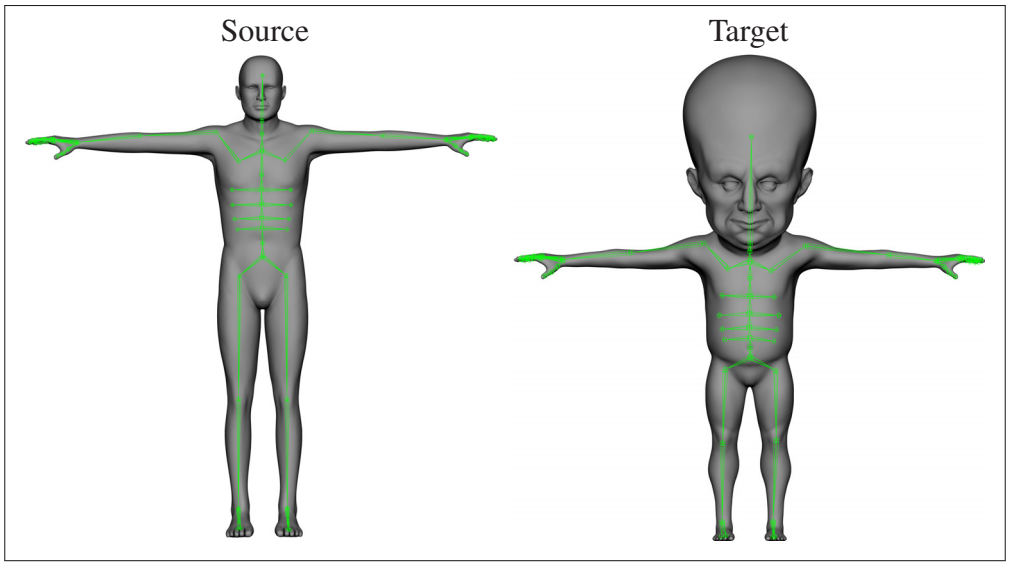



Figure 4.8 The Man and the Macrocephalic characters with a skeleton consisting of 75 joints.

## 4.2 Joint Position (*PA* vs. Energy Minimization)

As it was discussed in Chap. 3.1.2 the threshold for weighted vertices value has a great impact on the results for skeleton retargeting using *PA*. Considering only the vertices with high skinning weight values limits the point cloud of weighted vertices, but not any high threshold yields good results. Fig. 4.9 shows some skeleton retargeting examples using *PA* with different thresholds for weighted vertices values. As it can be seen in the figure, the $Threshold = 0.01$ gives better results than $Threshold = 0.3$, and $Threshold = 0.3$ is still better than $Threshold = 0.7$. The threshold should be a value which limits the weighted vertices, so aligning the point cloud of weighted vertices of the source to the target's becomes more meaningful. Thus, extremely limiting the point clouds leads to inaccurate results which yields wrong skeleton transferring (look at the shoulder joints of all characters in Fig. 4.9 where the threshold is set to 0.7 and compare it to where the threshold is set to 0.01).

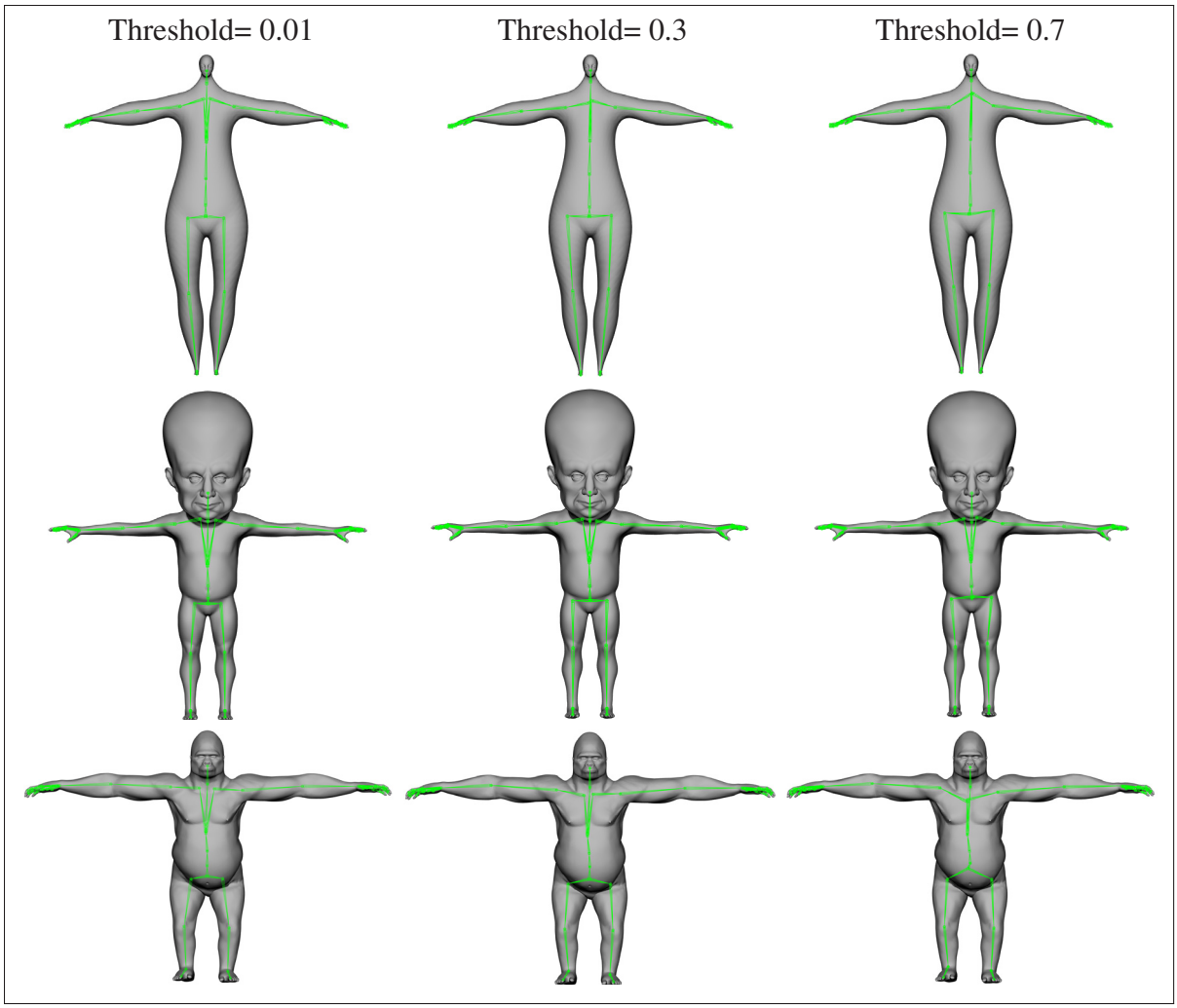



Figure 4.9 The impact of different thresholds on the results obtained using *PA*.

Among the two approaches that were used for retargeting the joint positions within the target skeleton, using *PA* was faster than using energy minimization. Fig. 4.10 shows the time spent for retargeting the skeleton using energy minimization and *PA*. Note that *PA* is nearly five times faster than energy minimization. Furthermore, implementing the *PA* is simpler than energy minimization because it avoids the creation of the linear system of partial derivatives required for the energy minimization approach. However, using *PA* is slightly less precise than using energy minimization according to various tests ran by a professional master student (refer to Appendix. II). These tests were performed for transferring the animation setup from a character

back to itself ($S_1 \rightarrow S_2 \rightarrow S_3 \rightarrow S_4$) as well as transferring a source character back and forth to a target character ($S_1 \rightarrow T_1 \rightarrow S_2 \rightarrow T_2$). Although the *PA* showed more differences (errors) in the matter of stability, the errors are still minor. Moreover, retargeting the skeleton using *PA* has the advantage of being able to handle out of the mesh joints as it does not rely on the joint's weighted vertices directly to express the joint position. Thus, to retarget an out of the mesh source joint or a joint with no related weighted vertices, *PA* can be used with the transformation of the parent joint.

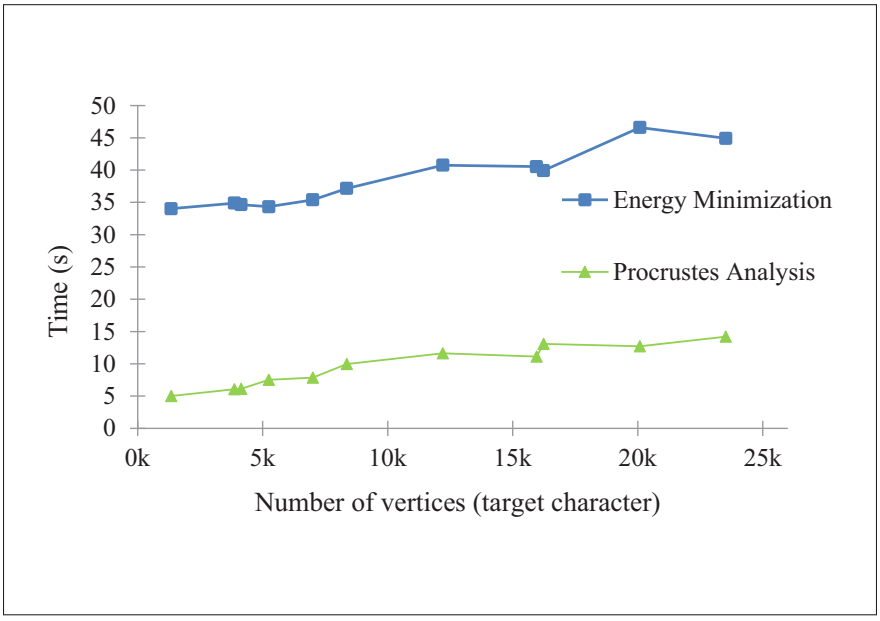



Figure 4.10 Computation time for skeleton retargeting using energy minimization and *PA* from the Man character as the source to all of the other eleven characters. The diagram is based on the target characters number of vertices.

Furthermore, for the same threshold of skinning weight values set to 0.05, if applied on both energy minimization and *PA*, the difference between these two approaches is notable (shown in Fig. 4.11). As it can be seen in Fig. 4.11 the spines of the Bulk and the Gorn characters after skeleton transferring using *PA*, even before applying the spine alignment, are more straight than the spines when transferred using energy minimization. Also, the skeletons seem

to be more symmetrical even without any mirroring while using *PA* compared to using energy minimization.

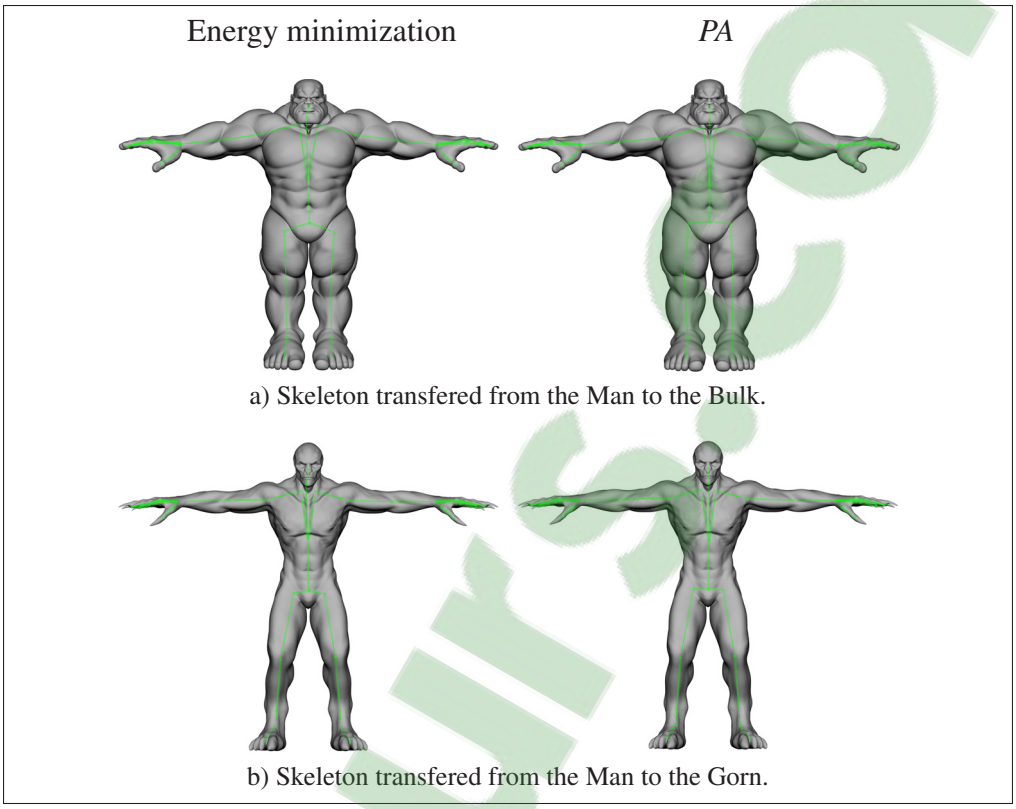



Figure 4.11 Skeleton retargeting from the Man as the source to the Bulk (top row) and the Gorn (bottom row) using energy minimization and *PA*. The threshold for the skinning weight value is set to 0.05 for both approaches.

## 4.3 Spine Alignment

As it was discussed in Chap. 3.1.3, the *spine alignment* process can be done using both the energy minimization approach and *PA*. Fig. 4.12 and 4.13 show the Elf and the Dwarf before and after *spine alignment* using *PA*. Fig. 4.14, 4.15, and 4.16 show the Bulk, the Gorilla, and the Sumo respectively, before and after *spine alignment* using energy minimization. The results obtained from using energy minimization and *PA* for the spine alignment are very similar and close in most cases. Thus, here different examples using either energy minimization or *PA* are presented as the visual examples of these approaches.

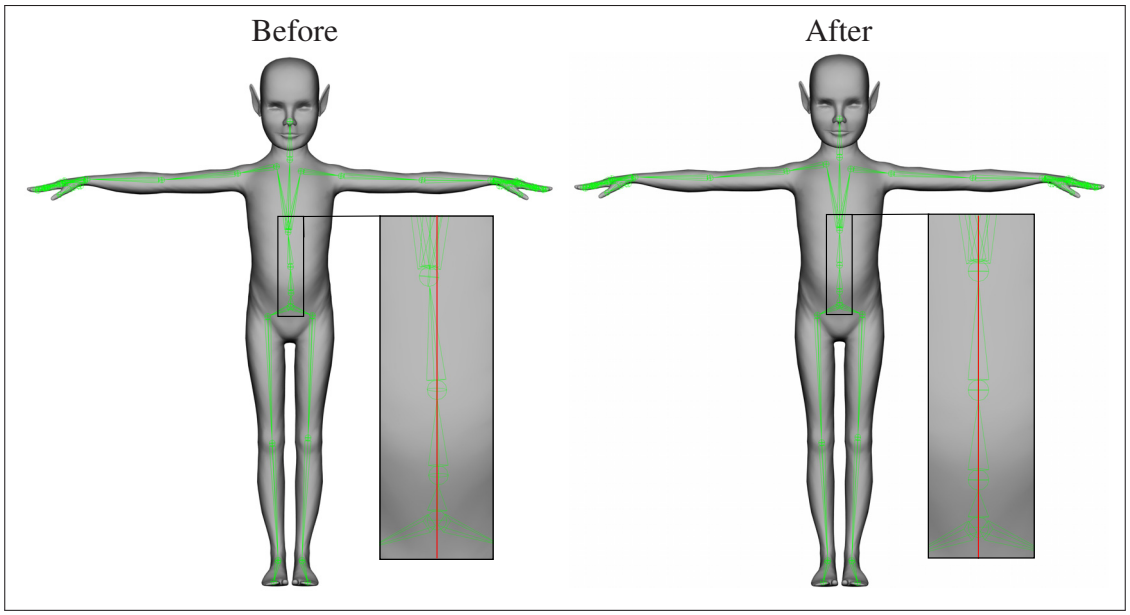



Figure 4.12 The Elf character before and after spine alignment. The joints transfer and *spine alignment* both were done using *PA*.

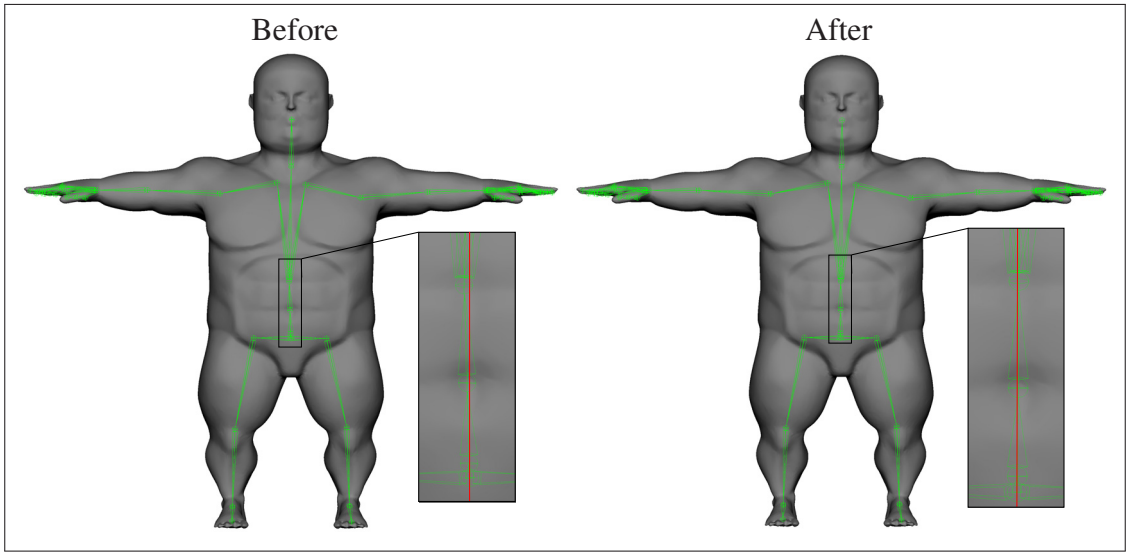



Figure 4.13 The Dwarf character before and after spine alignment. The joints transfer and *spine alignment* both were done using *PA*.

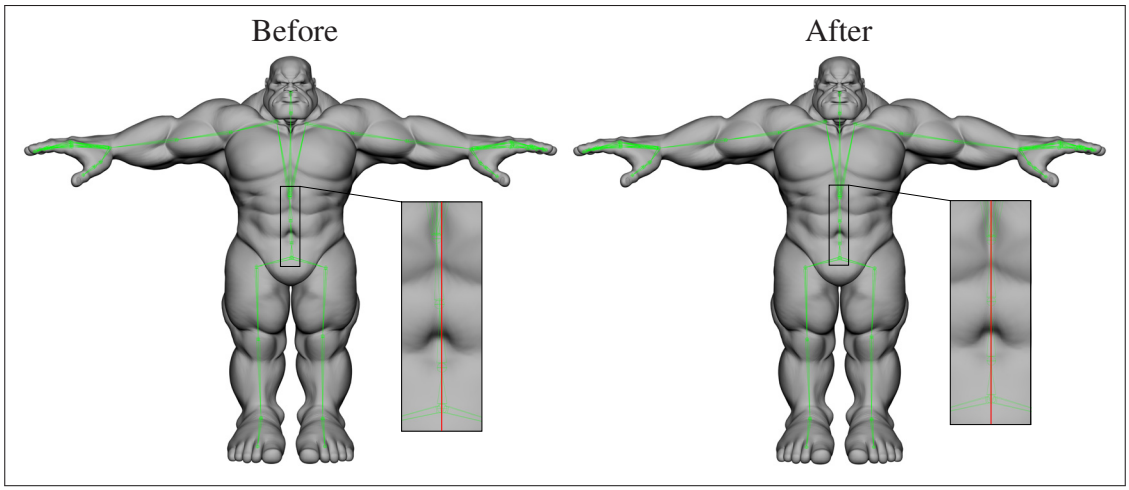



Figure 4.14 The Bulk character before and after spine alignment. The joints transfer and *spine alignment* both were done using energy minimization.

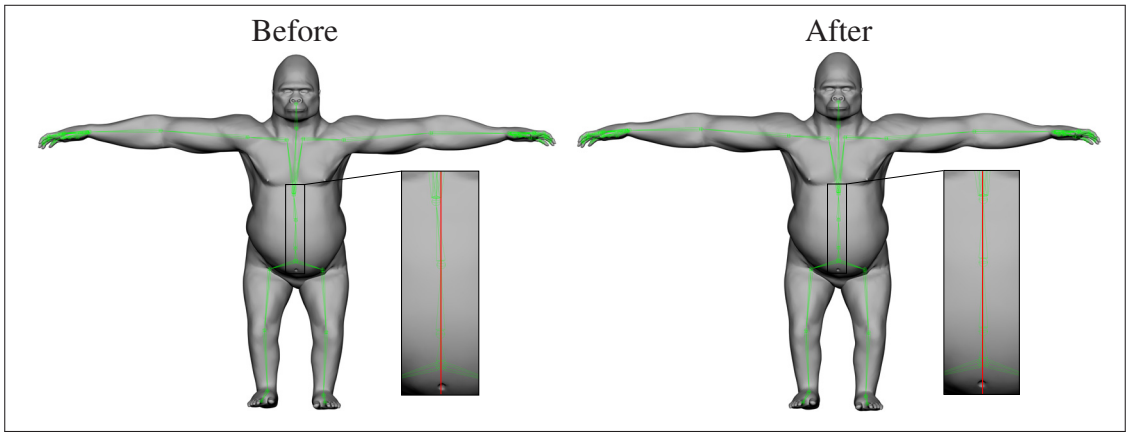



Figure 4.15 The Gorilla character before and after spine alignment. The joints transfer and *spine alignment* both were done using energy minimization.

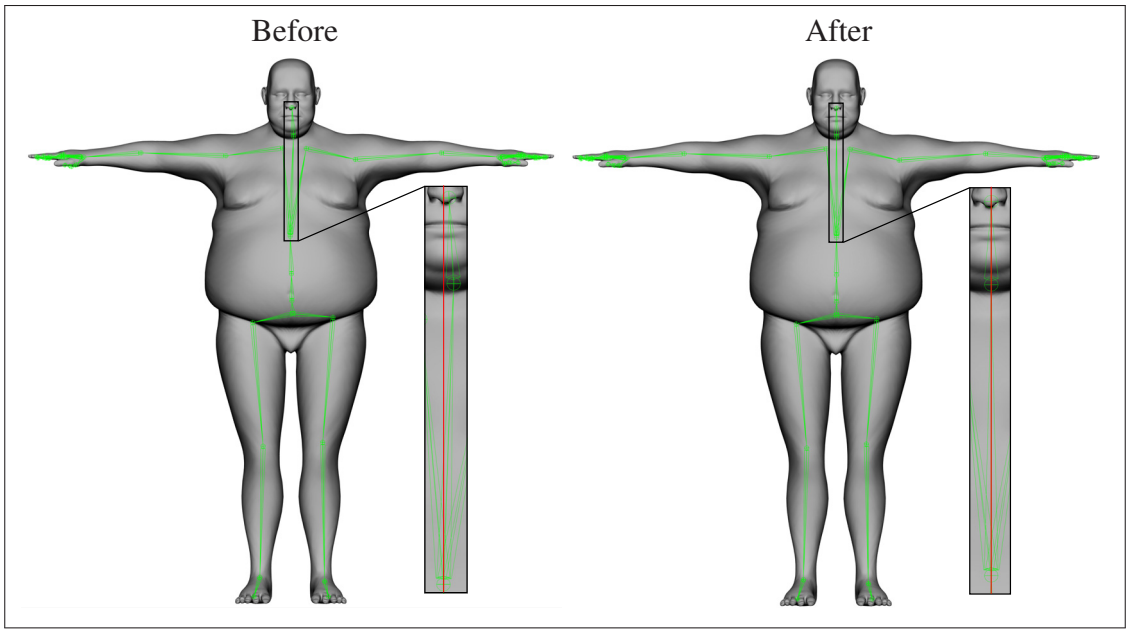



Figure 4.16 The Sumo character before and after spine alignment. The joints transfer and *spine alignment* both were done using energy minimization.

Furthermore, as it can be seen in Fig. 4.17 the target character shows less artifacts related to the spine joints within the mesh deformation after applying the spine alignment.

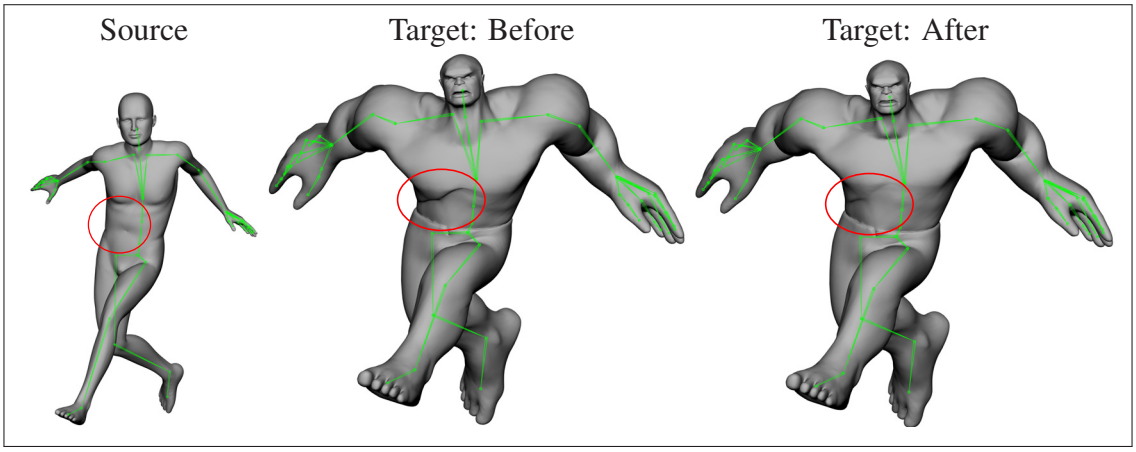



Figure 4.17 The Hulk character before and after spine alignment within the mesh deformation. The artifacts in the abdomen area related to spine joints have considerably decreased after applying the spine alignment.

Spine alignment using energy minimization yields the same $x$ coordinate for all of the spine joints. Thus, it leads to perfect results for all the characters with a straight abdomen. On the other hand, the results obtained from *spine alignment* using *PA*, do not lead to perfect results for all cases. An example of this case is shown in Fig. 4.18, where the whole spine is shifted further than the symmetry plane when using *PA*. In contrast, it can be seen that *spine alignment* using energy minimization shows more accurate results with spine joints exactly lying on the same $yz$ plane and it preserves the $x$ coordinates of the spine joints. However, the advantage of using *PA* for the *spine alignment* over using energy minimization, is that in all cases, it is able to compute the results much faster.

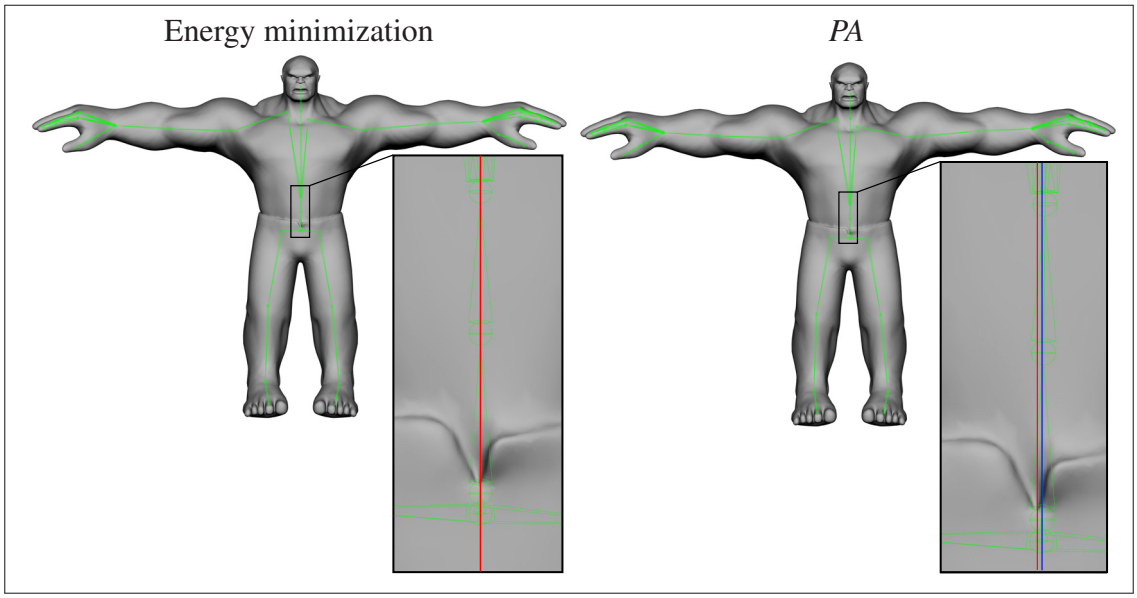



Figure 4.18 The skeleton is transferred from the Man to the Hulk and the *spine alignment* is applied to its spine joints using both energy minimization (left side of the figure) and *PA* (right side of the figure). In the *spine alignment* using energy minimization, the $yz$ plane which the spine joints are lying on, is highlighted in red. In the right figure, which shows the *spine alignment* using *PA*, the same $yz$ plane is highlighted in red, and the plane which the root joint is lying on when using *PA* is highlighted in blue. The shift in the spine joints set is notable here.

## 4.4 Mirroring

According to Chap. 3.1.4, mirroring is a process which symmetrizes the target skeleton according to a symmetry plane given by the user. Thus, applying such an approach on meshes which do not hold a symmetric geometry can not benefit from mirroring. Fig 4.19 shows an example of a character with asymmetric geometry, onto which mirroring does not lead to a correct result. Fig. 4.20 to 4.23 show several examples of 3D characters before and after applying the mirroring. As can be seen in Fig. 4.20 to 4.23, the mirroring operation greatly improves the results of animation setup transfer when the mesh is symmetric.

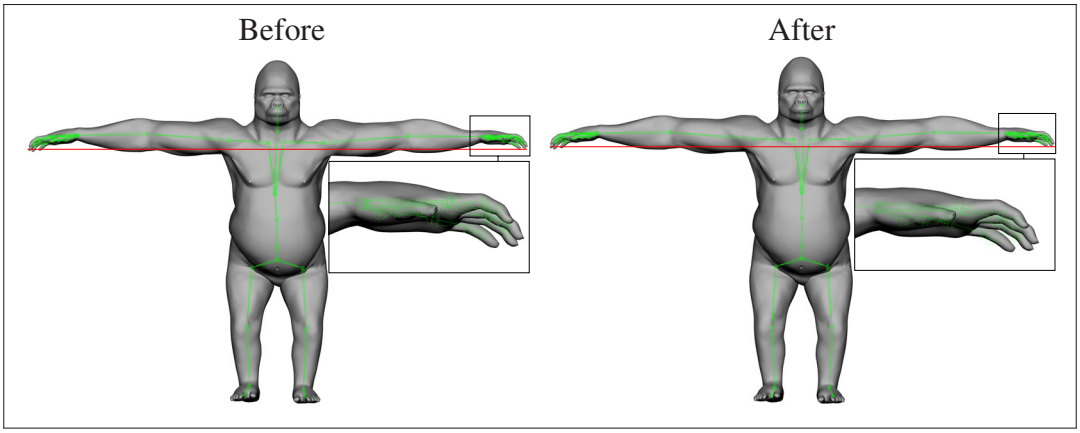



Figure 4.19 The Gorilla character before and after mirroring combined with spine alignment. The result of mirroring is worse because of the asymmetric geometry.

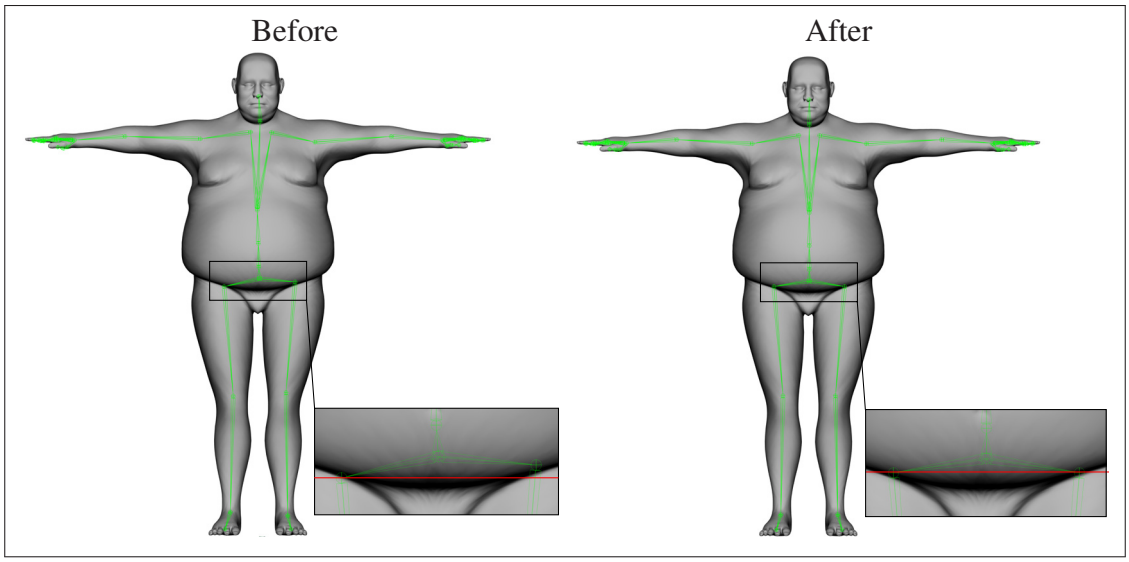



Figure 4.20 The Sumo character before and after mirroring its joints combined with spine alignment. The skeleton transfer and *spine alignment* both were done using energy minimization.

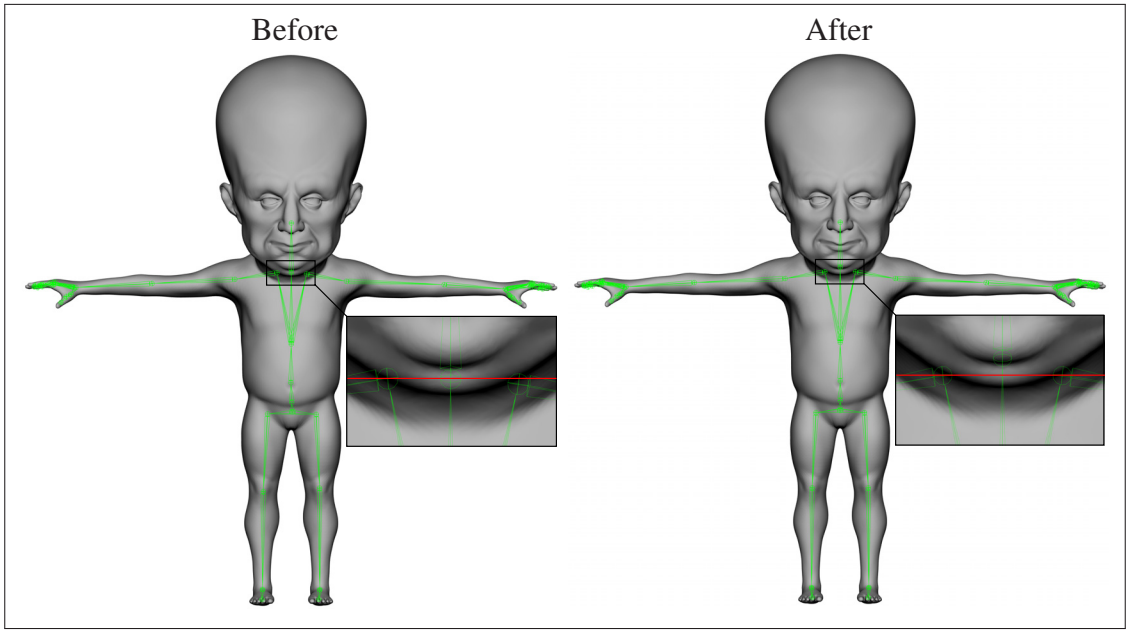



Figure 4.21 The Macrocephalic character before and after mirroring its joints combined with spine alignment. The skeleton transfer and *spine alignment* both were done using energy minimization.

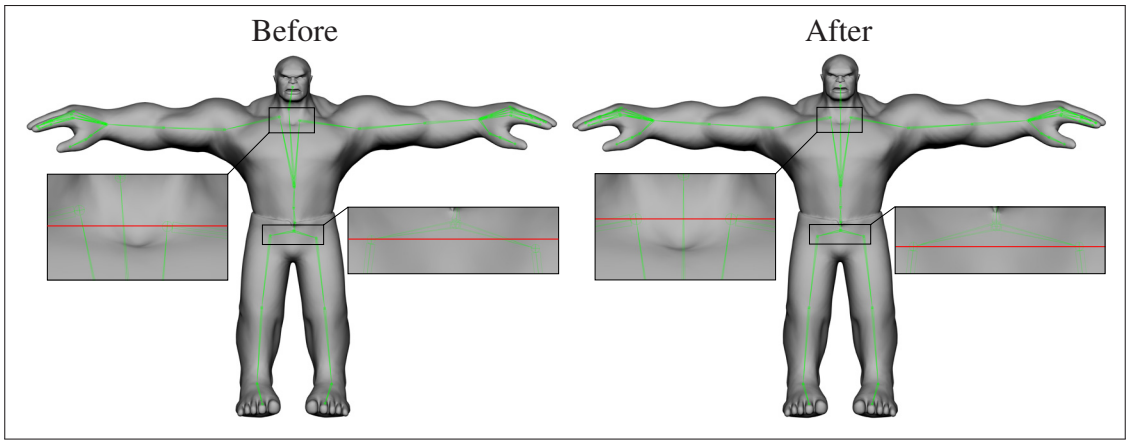



Figure 4.22 The Hulk character before and after mirroring its joints combined with spine alignment. The skeleton transfer and *spine alignment* both were done using energy minimization.

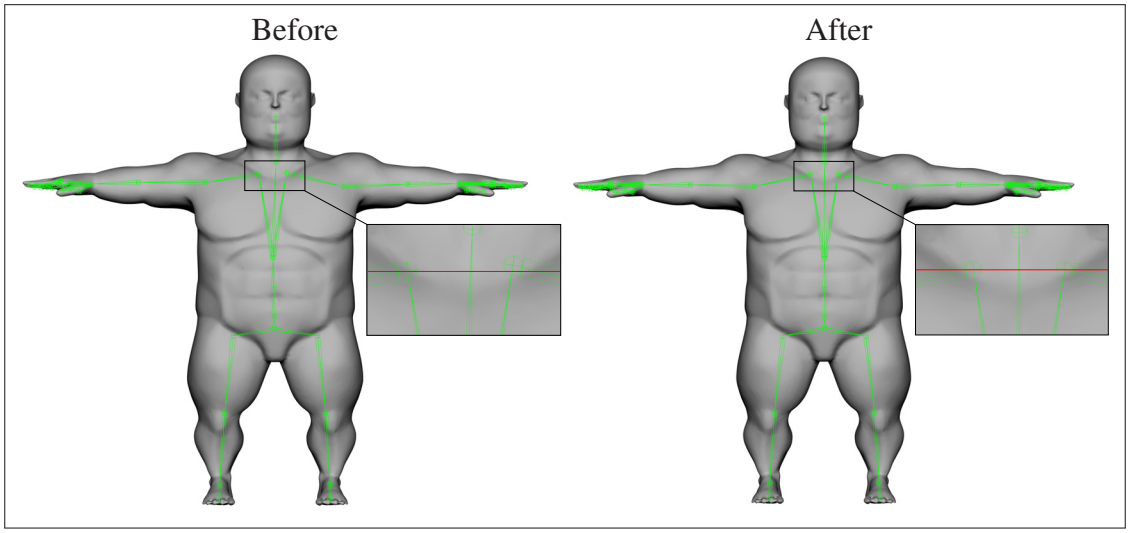



Figure 4.23 The Dwarf character before and after mirroring its joints combined with spine alignment. The skeleton transfer and *spine alignment* both were done using energy minimization.

## 4.5 Pose Normalization

The pose normalization shows its effect within the animation process. In Fig. 4.24 to 4.27, the same animation was applied to the Bulk, Gorilla, Curve, and Hulk characters. Some of the differences in limbs orientations before and after pose normaliztion are highlighted in red. The results shown in Fig. 4.24 to 4.27 verify the influence of the pose normalization on the limbs orientation. This approach significantly improves the results of animation setup transfer, as it modifies the target's limbs orientation so they follow the source's.

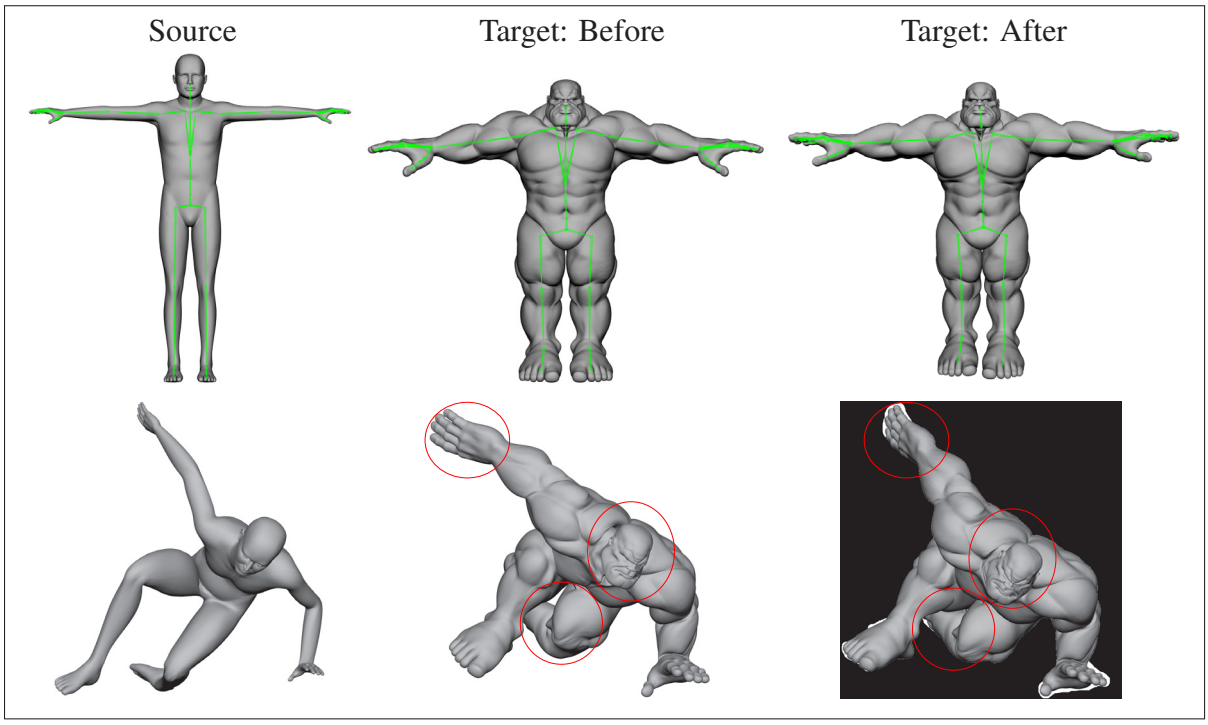



Figure 4.24 The Man character as the source and the Bulk character as the target, before and after applying the *pose normalization* in the first row. The second row shows the source and target within an animation, which is applied to both. Note the differences that *pose normalization* makes on the limb orientation within the animation.

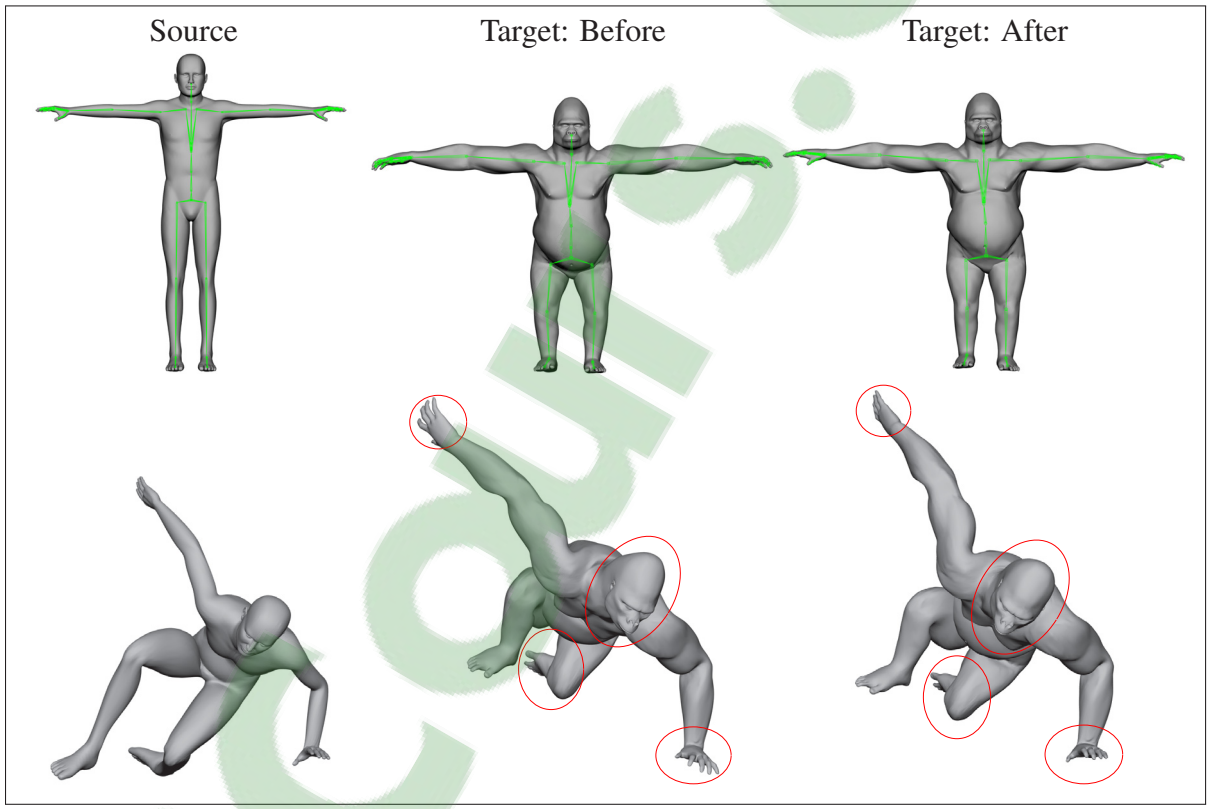



Figure 4.25 The Man character as the source and the Gorilla character as the target, before and after applying the *pose normalization* in the first row. The second row shows the source and target within an animation, which is applied to both. Note the differences that *pose normalization* makes on the limb orientation within the animation.

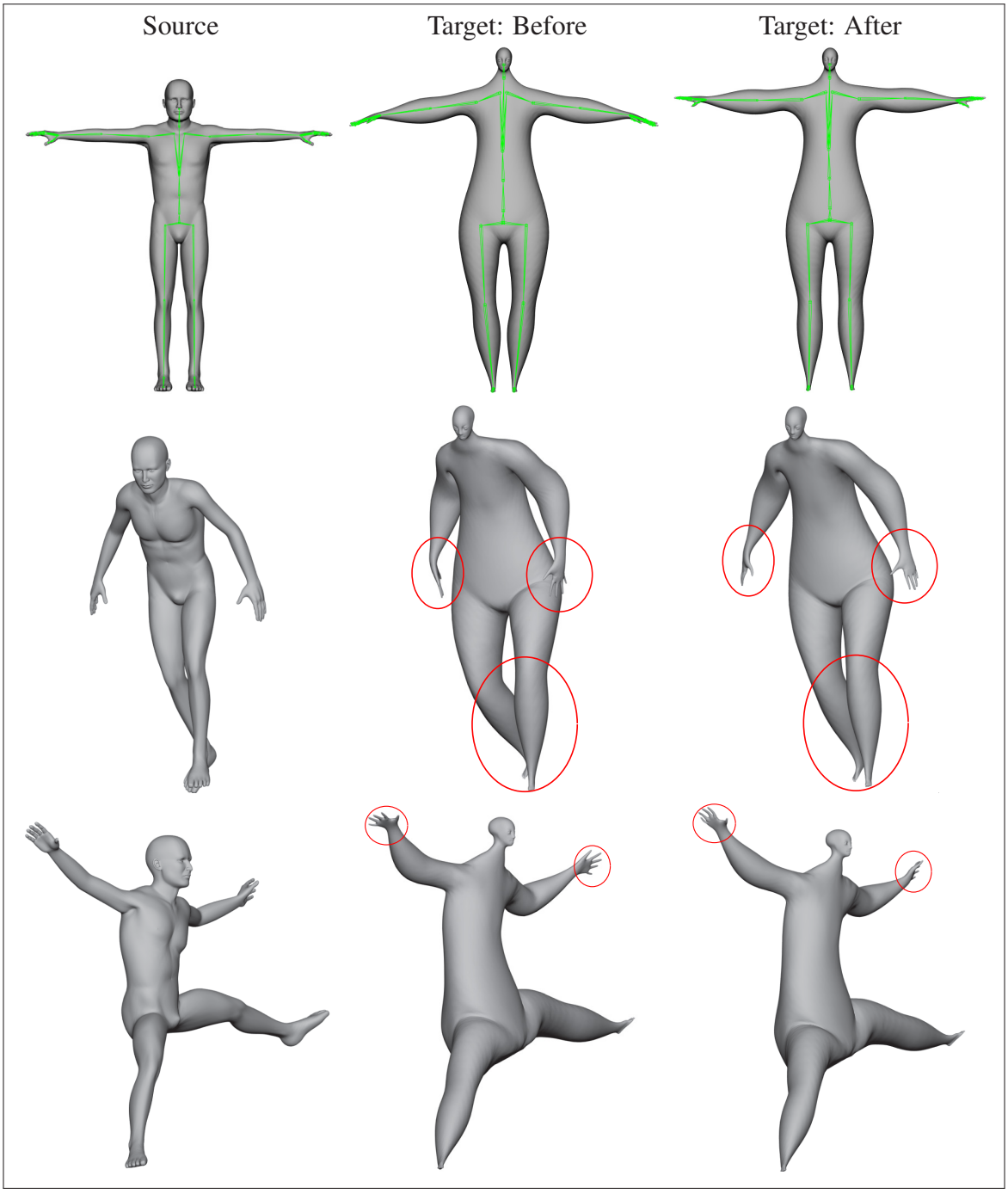



Figure 4.26 The Man character as the source and the Curve character as the target, before and after applying the *pose normalization* in the first row. The second and third rows show the source and target within an animation, which is applied to both. Note the differences that *pose normalization* makes on the limb orientation within the animation.

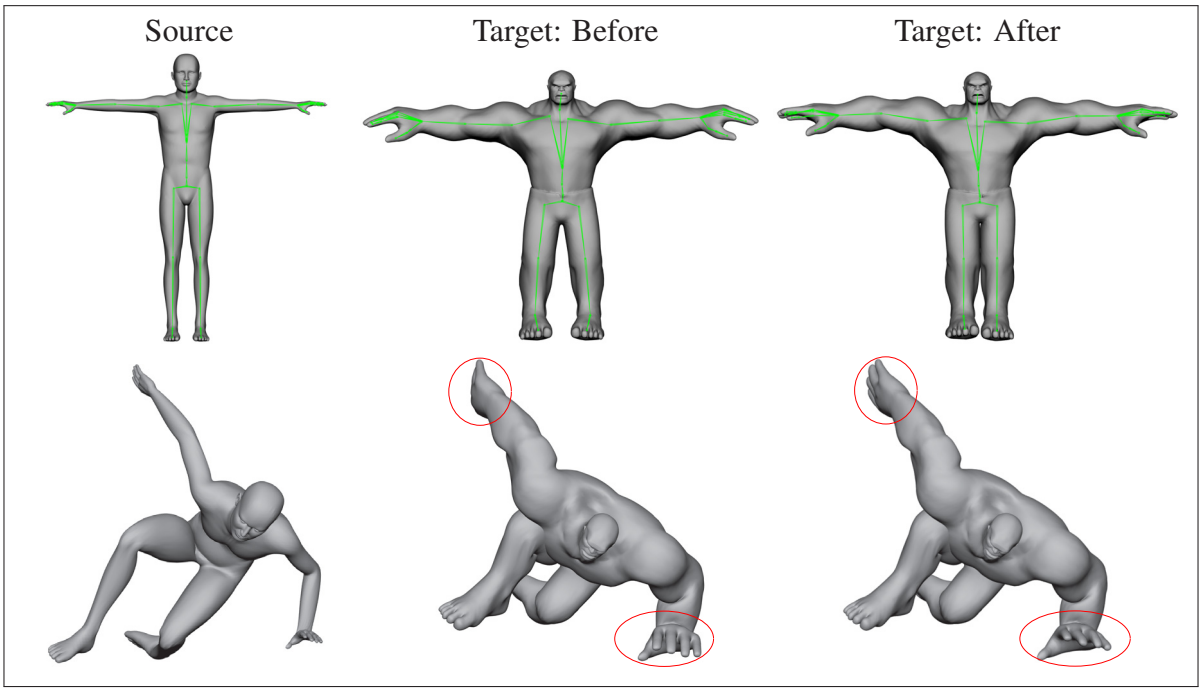



Figure 4.27 The Man character as the source and the Hulk character as the target, before and after applying the *pose normalization* in the first row. The second row shows the source and target within an animation, which is applied to both. Note the differences that *pose normalization* makes on the limb orientation within the animation.

# CHAPTER 5

# DISCUSSION

This chapter presents limitations of the approaches and potential solutions to overcome them. As it was mentioned in Chap. 4, twelve characters (refer to Fig. 4.1 and Tab. 4.1) were used to test the pipeline of the "Animation Setup Transfer" and the optimization approaches proposed in Chap. 3. All of the tests and results mentioned in this dissertation are for the implementation running on a 2.6 GHz Intel Core-i7 computer with 16 GB of memory. The pre-process of putting markers on source and target meshes was done in an application which is implemented using C++. The implementation of the geometric correspondence and skeleton transfer was performed in Matlab to benefit from its available linear solver and sparse matrix systems. The implementation of weight transferring was done using the Autodesk Maya python interface, taking advantage of the reverse mapping from the target mesh to the source mesh. Then, in Autodesk Maya, the results of the previous steps were used to visualize the whole pipeline in animation.

As part of this dissertation, the quality of the mesh deformation when animating characters using the skin weights transferred within the process of "Animation Setup Transfer" was compared to the deformation obtained from the use of two common automatic skin binding methods by Baran and Popović (2007) and Dionne and de Lasa (2013). While the results of the mesh deformation using these automatic binding methods were satisfactory, several issues were still identified which do not appear when manually produced skin weights were retargeted using "Animation Setup Transfer". Fig. 5.1 shows some of these issues in comparison with our approach.

The generality of the approaches enables them to be applied on a variety of 3D characters with different geometries and topologies. Furthermore, the advantage of having results with less artifacts and issues over other available methods such as automatic binding methods (Fig. 5.1) and also the methods already being used by software such as Maya to copy the skin weights

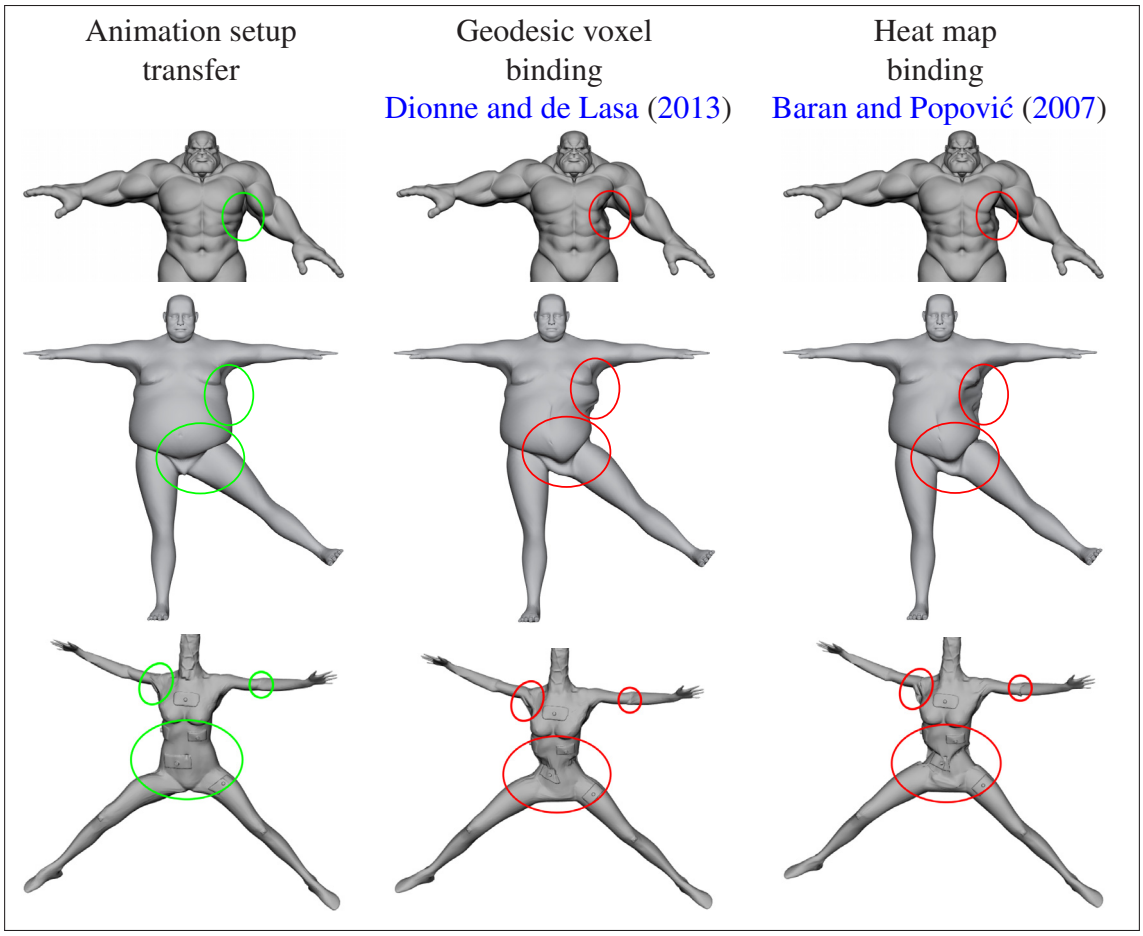



Figure 5.1 Mesh deformation using the transferred weights by the "Animation Setup Transfer" method compared to weights generated by automatic binding methods.

from one character to the other (e.g. Ray casting by Miller *et al.* (2010)), make our approaches more trustworthy.

## 5.1 Limitations

One of the major limitations of "Animation Setup Transfer" is the *Elastiface* method which is used to compute the geometric correspondence. Although this method showed better results comparing to other methods such as *Mobius Voting* by Lipman and Funkhouser (2009), *Blended Intrinsic Maps* by Kim *et al.* (2011), and *Deformation Transfer* by Sumner and Popović (2004) according to our requirements for non-isometric meshes, it still requires the source and

target meshes to be in similar poses. *Elastiface* is only able to handle slight differences in the poses as it can be seen in our examples. However, it should be mentioned that our approaches do not rely on a specific geometric correspondence method. Thus, should one use any other geometric correspondence method which is able to handle more drastic pose differences, it does not affect other stages of the "Animation Setup Transfer" process. This advantage is tightly related to the joint retargeting approach which is introduced in this thesis. Fig. 5.2 from the paper of Avril *et al.* (2016) shows the same target mesh in very different poses into which the skeleton is still transferred correctly.

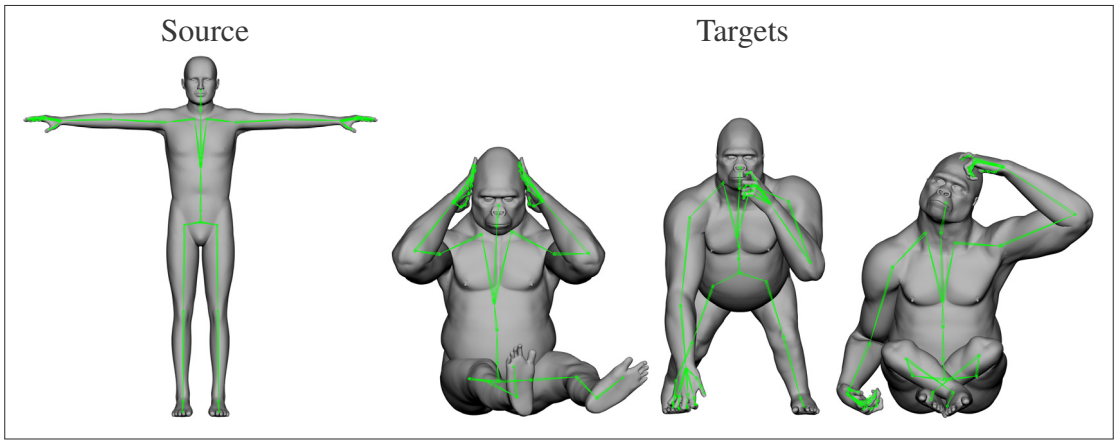



Figure 5.2 The Man skeleton is transferred to the Gorilla set in different poses. The geometric correspondence is computed using *Elastiface* from the Man to the T-pose Gorilla. Then, as the meshes share the same geometry and number of vertices and each vertex on the source corresponds to a barycentric coordinate on the target mesh, the same geometric correspondence was used to complete this skeleton retargeting. It is notable that even with this variation in pose, the joints are correctly positioned within the target meshes.

As it was discussed in Chap. 3.1.1 the joint positioning can be done either using energy minimization or *PA*. Skeleton retargeting using energy minimization can only handle the joints which lie within the convex envelope of the source mesh itself. This assumption is held in all of the test skeletons in this project. It means that, for example those joints that are used by artists to have more control over the last knuckle of each finger and are put out of the mesh geometry, can not be considered in the process of skeleton retargeting using energy minimiza-

tion. The reason is that those joints do not have any skin weights and they are only used so there can be a bone connecting the last knuckle of the finger to the top of it. This limitation may not appear in the *PA* approach because in this part we are dealing with transforming each joint according to the transformation of the point cloud of its weighted vertices from source to target. Thus, if for each joint with no weighted vertices, the same transformation of its parent joint is considered, then the approach can also be applicable to this case.

Furthermore, within the pose normalization process for putting the meshes in a neutral pose, if the parent joint has more than one child joints which do not share the same position, this approach can only have one $\Delta rotation$ of the parent joint to correct the position of all of its children. This limitation is closely related to the pose normalization approach proposed in this dissertation. An example of this situation is shown in Fig. 5.3.

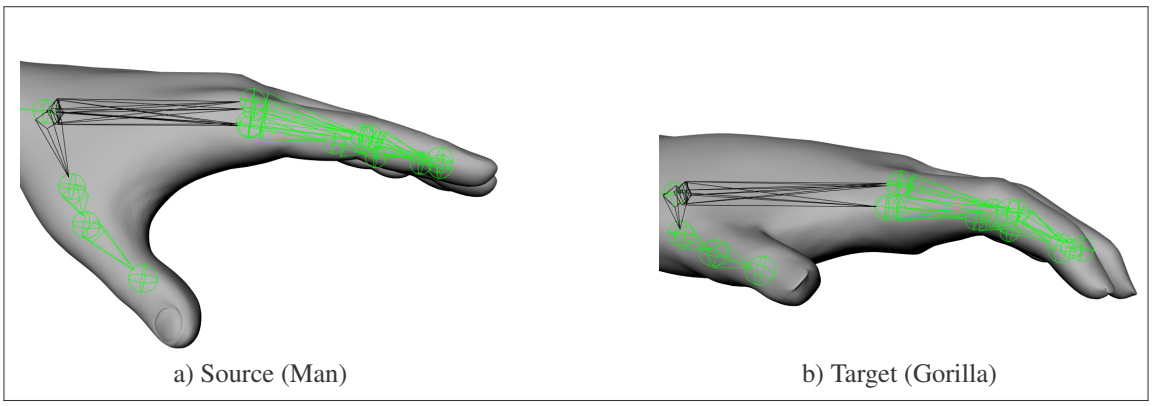



Figure 5.3 Parent joint (wrist) with several child joints (first knuckle of fingers). The position of each child joint is the result of a different translation in the parent's (wrist) local space (bones identified in black). While each child joint is correctly positioned regarding the pose of the target limbs by the skeleton retargeting approach using energy minimization (b), the pose normalization has a single Δrotation at the wrist which also transforms the knuckle joints as they are all connected by their parent joint. This prevents the bones in black to have an individual pose.

In this case, the bones highlighted in black can only be rotated together by the single wrist joint. As it can be seen in the Fig. 5.3, the angle between the thumb and the index finger in the Man's hand (Fig. 5.3a) is different from the same angle in the target's hand (Fig. 5.3b). Hence,

in the pose normalization process, the black bone of the target's thumb should be rotated by $\Delta rotation$ to correspond to the orientation of the thumb in the source. This rotation does not affect only the thumb's knuckle, but also it transforms the other knuckles in the target mesh by this same $\Delta rotation$. This means that both thumbs and fingers cannot be correctly aligned at the same time.

# CONCLUSION AND RECOMMENDATIONS

In the character animation, designing a suitable skeleton for a mesh and also rigging it according to its skeletal structure is a very time consuming task. "Animation Setup Transfer" presents an approach to transfer all the *animation setup* consisting of the skeleton and skin weights from a ready-to-animate character to one or many target characters. This dissertation is done as a part of this project which was launched in the Multimedia Laboratory of ÉTS. "Animation Setup Transfer" proposes a method using energy minimization for retargeting the skeleton (positioning the source joints within the target mesh). In this dissertation, in addition to explaining this method, a faster approach using *PA* is presented. To avoid non-aligned spine joints, a *spine alignment* approach is introduced using both energy minimization and *PA*. Furthermore, to take advantage of the symmetry of some meshes, a mirroring approach is proposed in order to symmetrize the skeleton according to a given symmetry plane. After transferring the skeleton, the orientation of each joint is transferred from the source to the target and their rotations are calculated to avoid inconsistency between the source and target within the animation process. Besides, a *pose normalization* approach is suggested to put the target meshes in the same pose as the source character (aligning the source and target characters by aligning their corresponding joints and limbs), which greatly eases the animation reuse.

The presented approaches are flexible, as they can handle character meshes with various topologies and morphologies. They were tested on a broad range of characters with different number of vertices, varying in topologies and morphologies. Different sets of skeletons were also successfully tested through the proposed skeleton transferring approaches which gives more freedom to artists. Moreover, the presented pipeline shows more reliable results comparing to common methods to create the skeleton and weights from scratch (either manually or using automatic methods), which need some post-steps to modify either the resulting skeleton or the skin weights. Thus, they can significantly reduce the time spent on preparing a character for animation. These approaches are not restricted to the method which computes the geomet-

ric correspondence, hence if another geometric correspondence method that can overcome the limitations of *Elastiface* is used, the proposed approaches would still work completely.

The results of the presented approaches suggest several future directions. First of all, it can be proved that in spite of energy minimization, using *PA* leads to transcending the limitation of out the mesh joints. In this way, transferring joint positions, even when they are positioned outside of the envelop of the weighted vertices becomes possible. Furthermore, there exist a hypothesis that if semantic joints (e.g. elbow and knee joints) are slightly moved in most cases, this can improve the quality of the mesh deformation and reduce the artifacts. Finding a meaningful formula to adjust the semantic joints could be another future direction.

# APPENDIX I

# GEOMETRIC CORRESPONDENCE

As it was discussed in Chap. 2 the first step for transferring the animation setup from a source to a target character is to compute a coarse geometric correspondence from the source vertices to points on the target. In the paper of Avril *et al.* (2016) the method of *Elastiface* by Zell and Botsch (2013) is chosen and implemented to be used for this purpose. In this section, we briefly explain the details of the *Elastiface* method in addition to the modifications that were applied to this method. The implementation of *Elastiface* was done by Quentin Avril, the post doctoral fellow, Donya Ghafourzadeh, and Srinivasan Ramachandran, two PhD students in the Multimedia Laboratory of ÉTS.

As it was discussed in Chap. 2, *Elastiface* proposes a method for establishing a geometric correspondence between a source and a target mesh. In spite of various approaches for obtaining the geometric correspondence, this method can handle non-isometric input characters. *Elastiface* extends linear NRR techniques in order to be able to handle varying input geometries and topologies. The whole method consists of: (1) setting markers, (2) transforming the source and target into smoothed base meshes (fairing step), (3) deforming the source mesh so that it matches the target using NRR techniques, and (4) extracting the vertex-to-point correspondence based on the closest locations between the deformed source and the target.

First some markers are manually placed on the semantic places of the source and target and linked together. Based on these manual correspondence specification, the registration initialized by aligning the source and target meshes using the best-matching similarity transformation (by Umeyama (1991)). After the alignment, the source and the target meshes are transformed into similar plain and feature-less shapes on which the robust correspondence is computed then. This step is called *fairing*. The fairing is applied to the meshes in order to remove their geometric details. Then, the corresponding markers are forced to coincide so a sufficient geometric similarity can be achieved. The correspondences are allowed to move to a certain degree in the next step in order to open up some geometrically complex and dense areas such as mouth and

nose. This transformation is done by optimizing the vertex positions of both the source ($M_{\mathscr{S}}$) and the target ($M_{\mathscr{T}}$) by minimizing the following energy function:

$$E(x_1^s,\ldots,x_n^s,x_1^t,\ldots,x_m^t) = \frac{\lambda_1}{\sum_i A_i^s + \sum_j A_j^t}\left[\sum_{i=1}^{n} A_i^s \left\|\Delta x_i^s\right\|^2 + \sum_{j=1}^{m} A_j^t \left\|\Delta x_j^t\right\|^2\right] \tag{A I-1a}$$

$$+ \frac{\lambda_2}{K}\sum_{k=1}^{K}\left\|r_k^s - r_k^t\right\|^2 \tag{A I-1b}$$

$$+ \frac{\lambda_3}{K}\left\|\frac{1}{2}(r_k^s + r_k^t - \frac{1}{2}(\bar{r}_k^s + \bar{r}_k^t))\right\|^2, \tag{A I-1c}$$

where $x_i^s$ and $x_j^t$ denote the vertices of the source and target mesh respectively, and $i = 1,\ldots,n$ and $j = 1,\ldots,m$ show the number of vertices in the source and target meshes. Eq. A I-1a minimizes the squared norms of per-vertex Laplacians $\Delta x_i$, weighted by their Voronoi areas $A_i$. Minimizing this term leads to as smooth as possible versions of the meshes. The second term (Eq. A I-1b) minimizes the deviation of corresponding marker positions. Hence, $r_k^s$ and $r_k^t$ are the barycentric combination of vertices $x_i^s$ and $x_j^t$ corresponding to the position of markers. The last term (Eq. A I-1c) is required to make the final answers unique and keep the average of the marker positions as close as possible to the average of their original positions ($\frac{1}{2}(\bar{r}_k^s + \bar{r}_k^t)$) before optimization.

The smoothed source and target meshes obtained from the fairing step have smoothed geometric details. Hence, a deformation-based NRR approach can easily be applied to them. The smoothed source mesh is first deformed onto the smoothed target mesh, then their resulting vertex correspondences are used as initial input guess for the original source and target meshes' registration. Eq. A I-2 shows the NRR step.

$$E(x_1^s,\ldots,x_n^s) = \frac{\mu_1}{\sum_i \hat{A}_i^s}\sum_{i=1}^{n}\hat{A}_i^s \left\|\Delta x_i^s - \Delta\hat{x}_i^s\right\|^2 \tag{A I-2a}$$

$$+ \frac{\mu_2}{n}\sum_{i=1}^{n}\left\|x_i^s - \hat{c}_i^t\right\|^2 \tag{A I-2b}$$

$$+ \frac{\mu_3}{K}\sum_{k=1}^{K}\left\|r_k^s - \hat{r}_k^t\right\|^2, \tag{A I-2c}$$

where $\hat{A}_i^s$ denotes the Vornoi area of the smoothed source mesh. Eq. A I-2a is the smoothness term which penalizes bending of the source model (i.e., change of curvature), measured by the displacement of the vertices Laplacian ($\Delta x_i^s - \Delta \hat{x}_i^s$) where $\hat{x}_i^s$ shows the vertex position on the smoothed source mesh. The fitting term (Eq. A I-2b) minimizes the distance of each source vertex $x_i^s$ to its closest point ($\hat{c}_i^t$) on the smoothed target mesh. The last term ensures that the markers $r_k$ remain at their positions. Next, the source mesh is deformed one more time using the NRR step to match the original target. In the last step of finding the geometric correspondence, the coarse vertex-to-point correspondence is computed. The final closest point correspondences ($x_i^s$, $\hat{c}_i^t$) computed on the smooth source and target meshes are used for matching the deformed source mesh to the original target mesh.

In order to solve these energy minimization problems, the linear system of partial derivatives equal to zero should be solved. We write the derivative of the equations in the form of $Ax = B$, so the matrix of constants ($B$) gets separated from the matrix of coefficients ($A$). This is done in Matlab where we first used the Biconjugate gradient (BiCG) method to solve this linear system. For some test cases, the results were far from the expected smooth meshes. At first it was assumed that these problems were related to the $\mu$ values used for the Eq. A I-2, so we ran a wide set of tests to check the results for different $\mu$ values. The hypothesis was that adjusting $\mu$ values would lead to correct answers, but after testing another method for solving the linear systems it was found that the BiCG method itself had difficulties converging to the solution minimize the energy function. To be able to correctly minimizing the energy function, the Matlab's *mlDivide* operator was used instead of BiCG. The Matlab mlDivide operator automatically selects an appropriate and exact solver based on criteria of the $A$ matrix. Fig. I-1 to I-4 show the results obtained from the NRR step using BiCG and *mlDivide* operators. The $\mu$ values are set the same for both computation techniques in each test case.

As it can be seen in Fig. I-1 to I-4, for those cases that BiCG does not converge to a good answer, the mlDivide is able to find a reasonable solution with the same $\mu$ parameters.

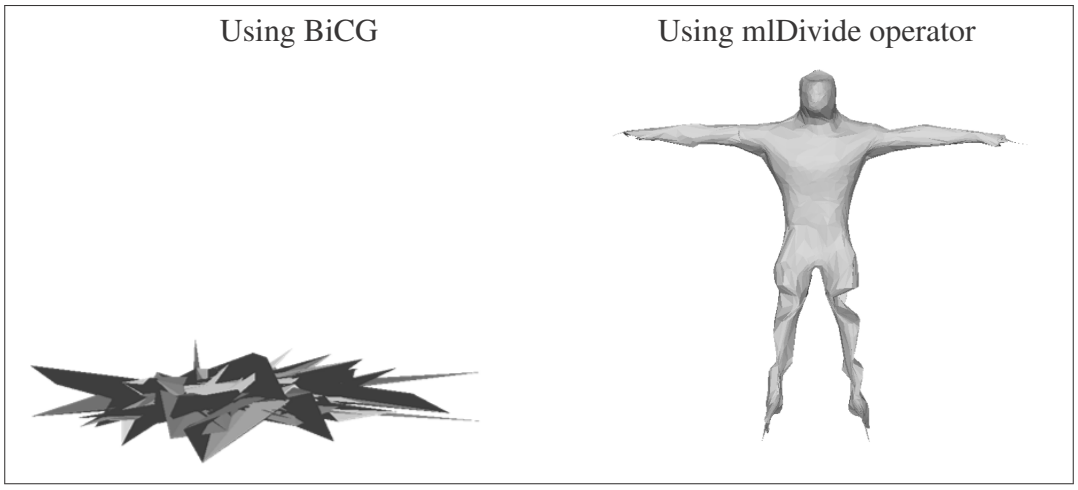



Figure-A I-1 The results from the NRR step using the BiCG and mlDivide approaches. The $\mu$ parameters are set to: $\mu_1 = 0.1$, $\mu_2 = 1$, and $\mu_3 = 0.1$.

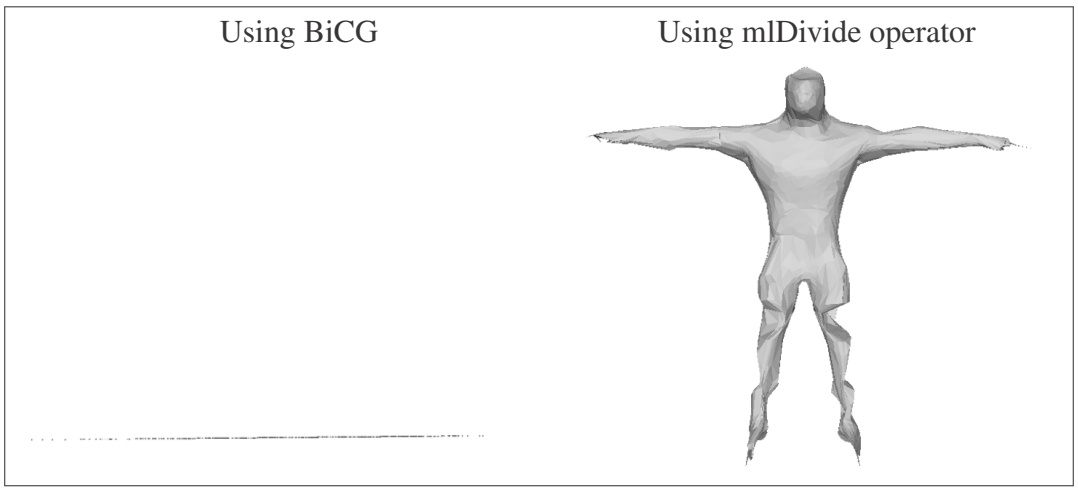



Figure-A I-2 The results from the NRR step using the BiCG and mlDivide approaches. The $\mu$ parameters are set to: $\mu_1 = 0.1$, $\mu_2 = 1$, and $\mu_3 = 1$.

In order to have a symmetric mapping between symmetric source and target meshes, a filtering approach for mirroring the geometric correspondence was proposed by the author. To this end, for each vertex on the completely symmetric source, the corresponding locations are found on the target mesh. The corresponding locations of each mirrored vertex on the source, are also found on the target. In order to have a symmetrical correspondence, an average between the initial corresponding location on the target and the mirrored position of the corresponding

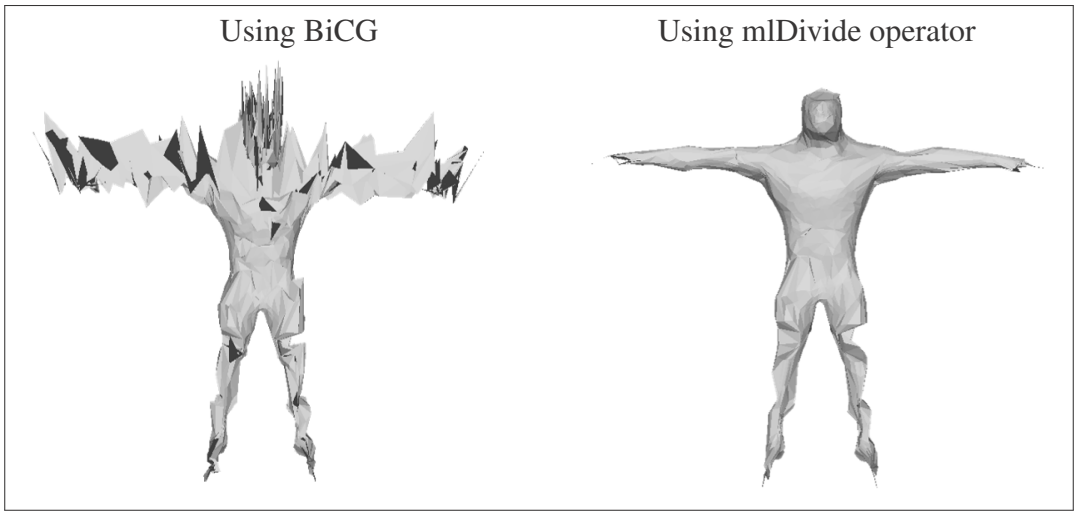



Figure-A I-3 The results from the NRR step using the BiCG and mlDivide approaches. The $\mu$ parameters are set to: $\mu_1 = 0.1$, $\mu_2 = 10$, and $\mu_3 = 10$.

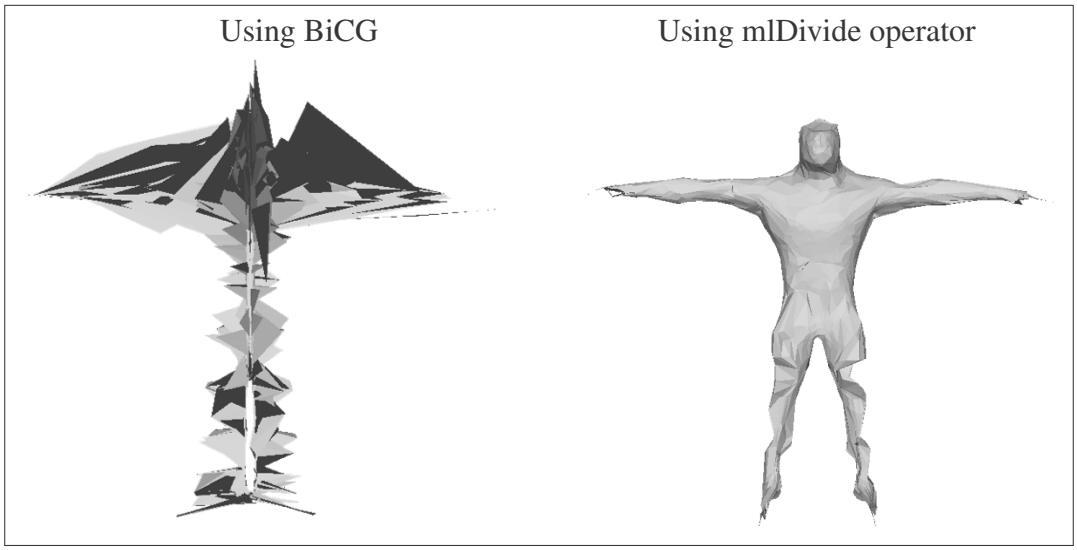



Figure-A I-4 The results from the NRR step using the BiCG and mlDivide approaches. The $\mu$ parameters are set to: $\mu_1 = 0.1$, $\mu_2 = 100$, and $\mu_3 = 100$.

mirrored vertex location is used instead. Fig. I-5 shows the mapping from the Macrocephalic character to the Man character, before and after applying this filter.

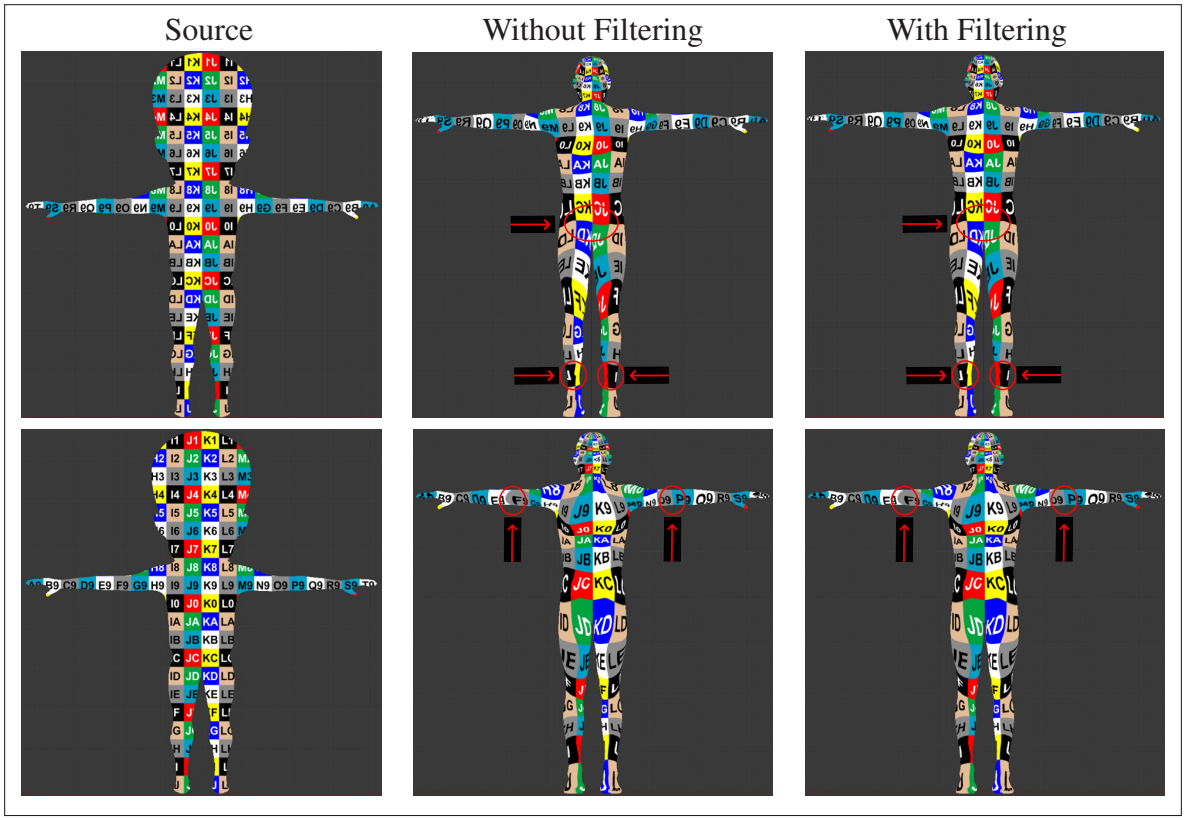



Figure-A I-5 The geometric correspondence from the Macrocephalic character to the Man, before and after applying the filter. The first row shows the front view and the second row shows the the back view. As it is highlighted in red, the results are considerably improved after applying the filter.

# APPENDIX II

## PERFORMANCE COMPARISON FOR THE STABILITY (EM VS. PA)

To evaluate the stability of the proposed approaches for skeleton retargeting, several animation setup transfer sets were performed. In one group of test cases, the skeleton and skinning weights were retargeted from the Man back to itself three times ($S_1 \rightarrow S_2 \rightarrow S_3 \rightarrow S4$). First the transfer was done using a direct vertex-to-vertex correspondence between the Man character and itself. Then, the same test was performed using the geometric correspondence computed by the *Elastiface* method. The animation setup transfers were tested using energy minimization and *PA*. The tests were performed by a professional master student. Fig. II-1 compares the stability of the skeleton transferring approach using energy minimization and *PA*. This test is done according to the perfect vertex-to-vertex correspondence from the Man back to itself. Fig. II-2 compares the same results but according to the geometric correspondence computed by *Elastiface* method. In the figures, the error for the energy minimization approach is often null. This is why the green bars often do not show up.

The second group of tests were done, transferring the animation setup back and forth from the Man to the Curve ($S_1 \rightarrow T_1 \rightarrow S_2 \rightarrow T_2$). These tests were performed by a professional master student. Fig. II-3 shows the comparison between the stability of the skeleton transferring using energy minimization and *PA*. Through both groups of test cases and in all of the animation setup transfers, using the *PA* method seems to bring slightly more errors (higher differences) than using energy minimization.

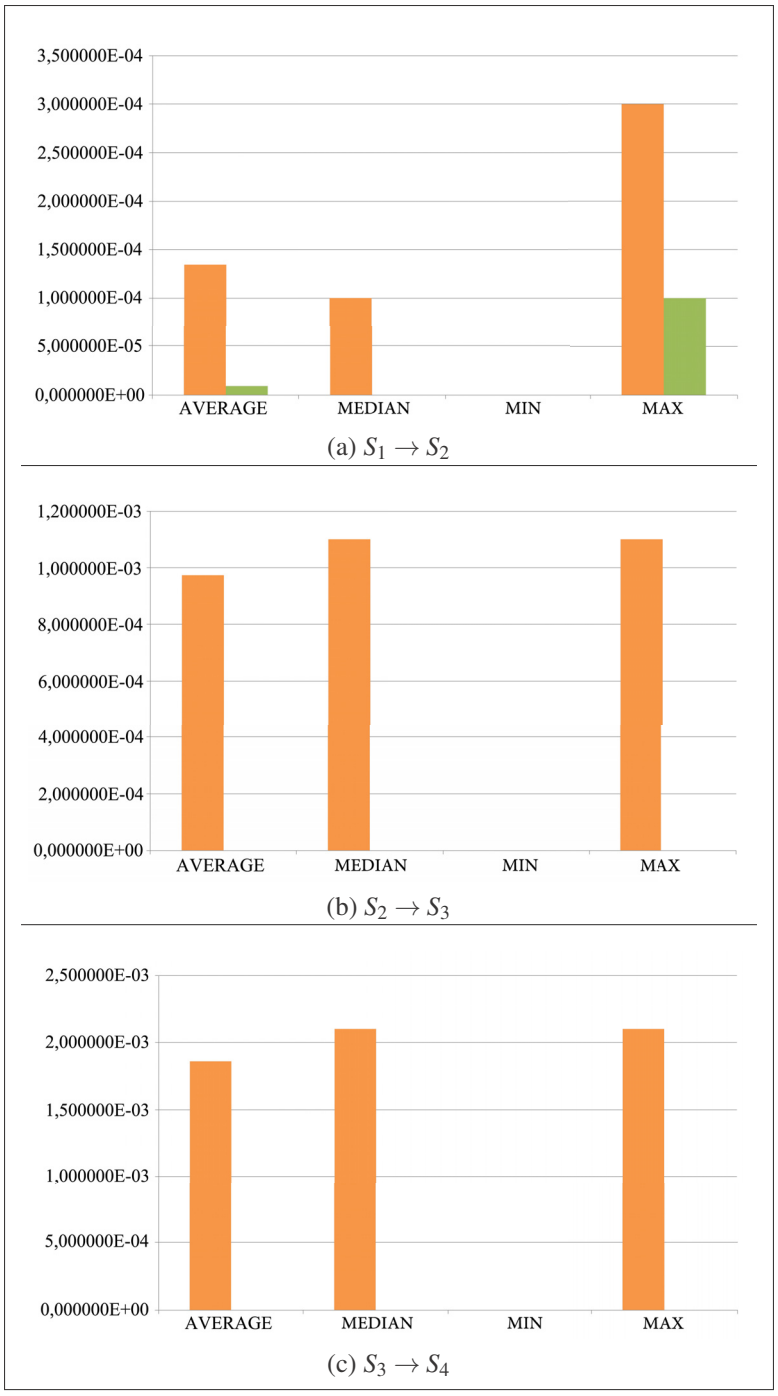



Figure-A II-1 Transfer the Man character's animation setup back to itself three times using a perfect vertex-to-vertex correspondence. The orange bars show the results using *PA* and the green ones show the results using energy minimization.

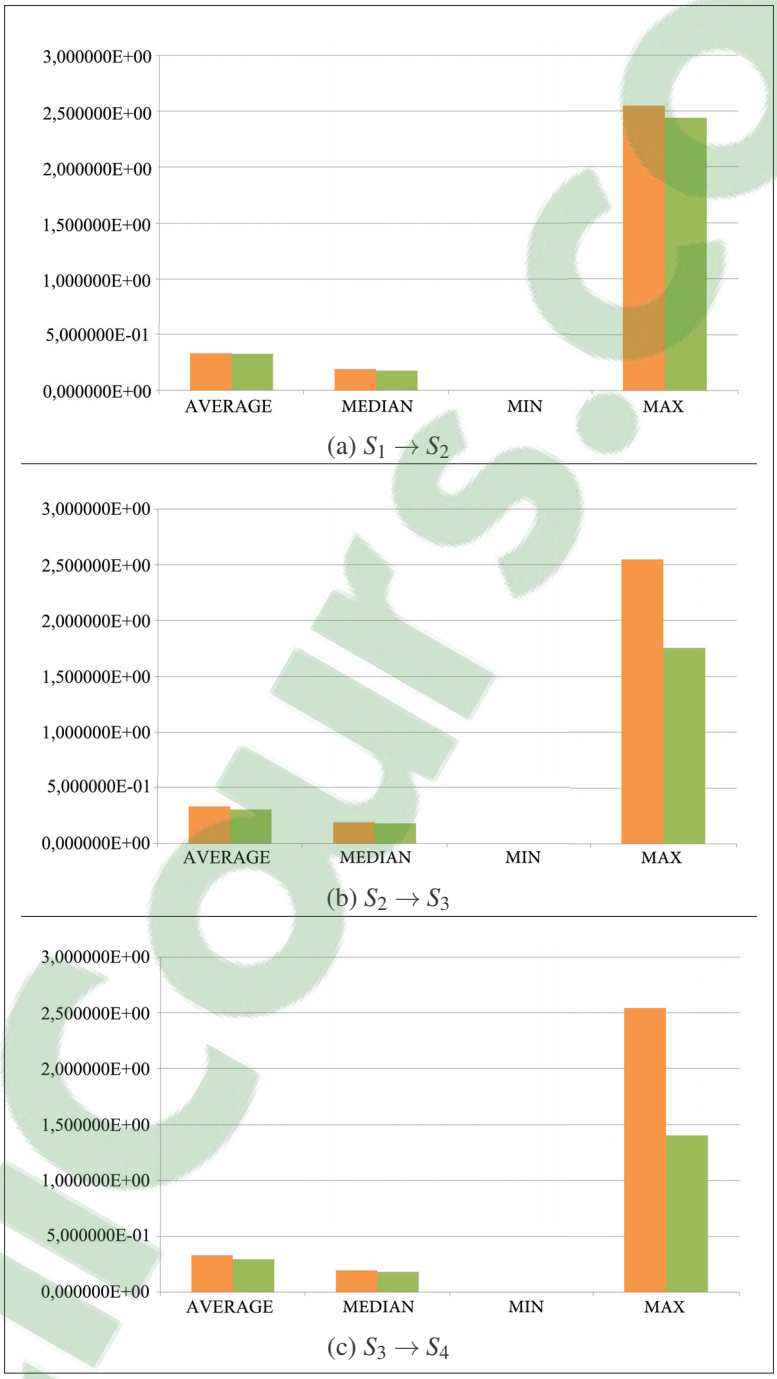



Figure-A II-2 Transfer the Man character's animation setup back to itself three times using *Elastiface* geometric correspondence. The orange bars show the results using *PA* and the green ones show the results using energy minimization.

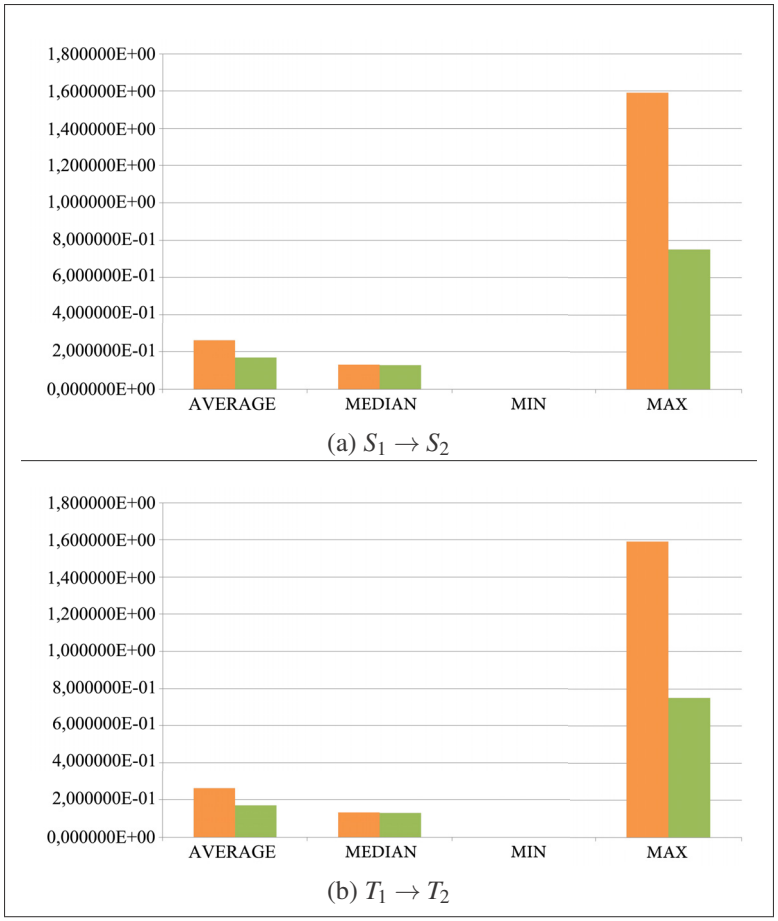



Figure-A II-3 Transfer the Man character's animation setup back and forth to the Curve. The orange bars show the results using *PA* and the green ones show the results using energy minimization.

## BIBLIOGRAPHY


Aigerman, N., R. Poranne, and Y. Lipman. 2014. "Lifted bijections for low distortion surface mappings". *ACM Transactions on Graphics (TOG)*, vol. 33, n° 4, p. 69.

Ali-Hamadi, D., T. Liu, B. Gilles, L. Kavan, F. Faure, O. Palombi, and M.-P. Cani. 2013. "Anatomy transfer". *ACM Transactions on Graphics (TOG)*, vol. 32, n° 6, p. 188.

Allen, B., B. Curless, and Z. Popović. 2003. "The space of human body shapes: reconstruction and parameterization from range scans". *ACM transactions on graphics (TOG)*, vol. 22, n° 3, p. 587–594.

Au, O. K.-C., C.-L. Tai, H.-K. Chu, D. Cohen-Or, and T.-Y. Lee. 2008. "Skeleton extraction by mesh contraction". *ACM Transactions on Graphics (TOG)*, vol. 27, n° 3, p. 44.

Avril, Q., D. Ghafourzadeh, S. Ramachandran, S. Fallahdoust, S. Ribet, O. Dionne, M. de Lasa, and E. Paquette. 2016. "Animation Setup Transfer for 3D Characters". *Computer Graphics Forum*, vol. 35, n° 2, p. 115–126.

Baran, I. and J. Popović. 2007. "Automatic rigging and animation of 3d characters". *ACM Transactions on Graphics (TOG)*, vol. 26, n° 3, p. 72.

Bronstein, A. M., M. M. Bronstein, and R. Kimmel. 2006. "Generalized multidimensional scaling: a framework for isometry-invariant partial surface matching". *Proceedings of the National Academy of Sciences*, vol. 103, n° 5, p. 1168–1172.

Chang, Y.-T., B.-Y. Chen, W.-C. Luo, and J.-B. Huang. 2006. "Skeleton-driven animation transfer based on consistent volume parameterization". *Advances in Computer Graphics*, p. 78–89.

De Aguiar, E., C. Theobalt, S. Thrun, and H.-P. Seidel. 2008. "Automatic Conversion of Mesh Animations into Skeleton-based Animations". *Computer Graphics Forum*, vol. 27, n° 2, p. 389–397.

Dionne, O. and M. de Lasa. 2013. "Geodesic voxel binding for production character meshes". *Proceedings of the 12th ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, p. 173–180.

Gower, J. C. 1975. "Generalized procrustes analysis". *Psychometrika*, vol. 40, n° 1, p. 33–51.

Gower, J. C. and G. B. Dijksterhuis, 2004. *Procrustes problems*. Number 30.

He, Y., X. Xiao, and H.-S. Seah. 2009. "Harmonic 1-form based skeleton extraction from examples". *Graphical Models*, vol. 71, n° 2, p. 49–62.

Kim, V. G., Y. Lipman, X. Chen, and T. Funkhouser. 2010. "Möbius transformations for global intrinsic symmetry analysis". *Computer Graphics Forum*, vol. 29, n° 5, p. 1689–1700.

Kim, V. G., Y. Lipman, and T. Funkhouser. 2011. "Blended intrinsic maps". *ACM Transactions on Graphics (TOG)*, vol. 30, n° 4, p. 79.

Le, B. H. and Z. Deng. 2012. "Smooth skinning decomposition with rigid bones". *ACM Transactions on Graphics (TOG)*, vol. 31, n° 6, p. 199.

Le, B. H. and Z. Deng. 2014. "Robust and accurate skeletal rigging from mesh sequences". *ACM Transactions on Graphics (TOG)*, vol. 33, n° 4, p. 84.

Li, H., R. W. Sumner, and M. Pauly. 2008. "Global Correspondence Optimization for Non-Rigid Registration of Depth Scans". *Computer Graphics Forum*, vol. 27, n° 5, p. 1421–1430.

Li, J. and G. Lu. 2011. "Skeleton driven animation based on implicit skinning". *Computers & Graphics*, vol. 35, n° 5, p. 945–954.

Lipman, Y. and T. Funkhouser. 2009. "Möbius voting for surface correspondence". *ACM Transactions on Graphics (TOG)*, vol. 28, n° 3, p. 72.

Miller, C., O. Arikan, and D. Fussell. 2010. "Frankenrigs: building character rigs from multiple sources". *Proceedings of the 2010 ACM SIGGRAPH symposium on Interactive 3D Graphics and Games*, p. 31–38.

Orvalho, V., P. Bastos, F. Parke, B. Oliveira, and X. Alvarez. 2012. "A facial rigging survey". *In 33rd Annual Conference of the European Association for Computer Graphics-EUROGRAPHICS*, vol. 51, p. 13–18.

Poirier, M. and E. Paquette. 2009a. "Rig retargeting for 3d animation". *Proceedings of Graphics Interface 2009*, p. 103–110.

Poirier, M. and E. Paquette. 2009b. "Methods for fast skeleton sketching". *SIGGRAPH 2009: Talks*, p. 54.

Seo, J., Y. Seol, D. Wi, Y. Kim, and J. Noh. 2010. "Rigging transfer". *Computer Animation and Virtual Worlds*, vol. 21, n° 3-4, p. 375–386.

Sumner, R. W. and J. Popović. 2004. "Deformation transfer for triangle meshes". *ACM Transactions on Graphics (TOG)*, vol. 23, n° 3, p. 399–405.

Umeyama, S. 1991. "Least-squares estimation of transformation parameters between two point patterns". *IEEE Transactions on pattern analysis and machine intelligence*, vol. 13, n° 4, p. 376–380.

Van Kaick, O., H. Zhang, G. Hamarneh, and D. Cohen-Or. 2011. "A survey on shape correspondence". *Computer Graphics Forum*, vol. 30, n° 6, p. 1681–1707.

Wang, X. C. and C. Phillips. 2002. "Multi-weight enveloping: least-squares approximation techniques for skin animation". In *Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on computer animation*. p. 129–138. ACM.

Wareham, R. and J. Lasenby. 2008. "Bone glow: An improved method for the assignment of weights for mesh deformation". *International Conference on Articulated Motion and Deformable Objects*, p. 63–71.

Zell, E. and M. Botsch. 2013. "ElastiFace: matching and blending textured faces". *Symposium on Non-Photorealistic Animation and Rendering*, p. 15–24.