EG UK Computer Graphics & Visual Computing (2019)
G. K. L. Tam and J. C. Roberts (Editors)

A Mesh Correspondence Approach for Efficient Animation Transfer

A. Moutafidou[1] and I. Fudos[1]

[1]Department of Computer Science & Engineering, University of Ioannina, Greece

Abstract

Animating a novel character not only needs a lot of effort and time but also requires the intervention of an experienced user. Moreover, the traditional animation pipeline for a set of characters can be a tedious and cumbersome process which often needs to be repeated several times to correct artifacts. We propose a user-friendly, semi-automated efficient method which is realized in two phases: (i) mesh correspondence, and (ii) skeleton and skinning transfer. We have developed a software tool for realizing the entire process without the need of third party software. We have improved the efficiency of the entire animation transfer process. To substantiate this, we provide a comparative performance evaluation with previous competent approaches.

1. Introduction

Creating an animated character from scratch is a cumbersome process that has to be realized by employing several tools in an iterative pipeline: (i) determine the overall kinematics (by mocap or physical based simulation), (ii) create the skeleton (rigging) (iii) determine the skinning weights and the joint areas (skinning), (iv) correct artifacts, (v) include local deformations or auxiliary bones and repeat (i)-(iv) if needed. An animation for a specific character consists of a set with geometric entities (vertices and faces), a skeleton, a many-to-many relation between vertices and skeleton bones, and the kinematics of the skeleton.

Even if we want to apply the same animation for different static characters (that comprise a mesh and one or more textures) we should repeat this entire process all over again. Therefore, it is of primary importance to have a tool to produce an animation for a new static character (also called the target) automatically by transferring the preexisting animation of another character (also called the source). This process is commonly referred to as *animation transfer* [AGR*16].

Geometry matching is a key step in the animation pipeline. So the first step of our approach is to establish a mapping of the geometric entities between source and target based on two sets of automatically generated marker points. The next step is to transfer the existing skeleton from source to target mesh and finally determine the skinning weights of the new model. Finally, the user may need to adjust the skinning weights to alleviate animation artifacts in joint areas.

We have introduced an automated approach to selecting marker point so as to expedite the animation transfer process. We have also improved significantly the efficiency of the overall animation transfer process as compared to previous approaches. Finally, we have conducted several experiments to substantiate the high quality and performance of our method.

2. Related Work

There are several previous works on animation transfer. [AGR*16] demonstrates a full animation transfer pipeline from a source to a set of targets [SLT*19]. This method requires significant effort from users to select manually pairs of marker points and uses several proprietary software tools during the process. We introduce an integrated platform for animation transfer with minimal user intervention.

[ZHS*05] focuses on geometry without creating a complete animation setup (without providing a skeleton or skinning weights in the target mesh).

Mesh correspondence is a key step in our approach. [vKZHCO11] discusses methods that are designed to compute correspondences between geometric shapes represented by triangle meshes. Moreover, the need to establish a correspondence even between meshes of different morphology suggest the use of marker points so that the user can formulate an arbitrary matching that captures his/her creative intent. In this case [ZB13] can provide a feasible solution. We provide a user friendly approach for selecting marker points built around a mesh clustering approach where the user has to provide correspondence pairs between two limited sets of representative points.

Skeleton fitting is the next step of our approach. There are methods that can achieve transfer between two characters via skeleton and weight fitting. For example [ACP03] presents a method that attains transfer between meshes with similar morphology because each joint is re-targeted based on three markers on the mesh.

[AHLG*13] on the other hand can achieve the re-targeting but only for characters that are abide to a predefined template.

The final fitting is based on the skinning weights. [WL08] presents a method that extracts the weights only from the skeleton. Other methods need a set of example poses to find the optimal weights [LD12, DATTS08].

We use a combination of mesh matching between source and target driven by the marker points that provide a correspondence of all geometry between the two meshes. Given that, the skeleton and weight fitting is then straightforward to derive.

(a) *Front view of models.* **(b)** *Back view of models.*

Figure 1: *P-center example of source and target models.*

3. Mesh Matching

An animation setup transfer is a method that given a source character and a target mesh can turn the target mesh into an animated character [AGR*16]. The source is an animated mesh with geometry, skeleton and skin. Targets are arbitrary meshes without additional information. The goal is to transfer all the necessary information via a pipeline which consists of automated and semi-automated steps.

3.1. *P-Center* Algorithm and Marker Points

Our approach requires a number of marker points to successfully align the source and target meshes. These marker points should preserve not only the most important areas of the models but also all rigid and deformable components. We present a method for finding a set of representative points that cover the entire mesh using the *P-center* clustering algorithm. *P-center* finds the minimum coverage distance until the entire model is covered [Gon85].

The algorithm works through an initialization step and $k-1$ "expanding" steps where k represents the number of marker points. In the first step all vertices are assigned to the first cluster B_1 and then we select the head of this cluster randomly. Accordingly, in every subsequent step we choose as head of the new cluster $head_j$ the vertex that has the maximal distance from the head of the cluster B_{j-1} that it currently belongs to. The new cluster B_j, will be populated with all vertices that are closer to $head_j$ as compared to the heads of all other clusters.

We use the *P-center* algorithm to obtain $2k$ marker points (heads of $2k$ clusters) for the source mesh, and k marker points (heads of k clusters) for the target mesh. Subsequently, the user for each of the $2k$ marker points of the source mesh picks a corresponding marker point on the target mesh. By having more candidate options on the source mesh, the users can establish a correspondence of the source and target mesh by choosing k pairs of marker points that best capture their design intent. In Figures 1aand 1b we can see a simple example of producing 50 representative points for the source and the target model.

3.2. User Study

We have performed a user study to evaluate the usability of the interactive marker selection method. We have measured: (i) the efficiency of our method in terms of the overall time needed by the users and (ii) the quality of the set of selected markers. Using "between groups design", we have asked one user group (group A) to use our tool and another user group (group B) to use the previous approach reported in [ZB13]. Each group consists of 16 subjects. Our role in group A, was to explain in users how the tool works and then ask them to perform some experiments. In group B, we also explained to users how they should pick marker pairs from the original meshes to cover uniformly all joints and asked them to select the marker points themselves. Note that in the first case the user chooses among a two sets of vertices produced by the *P-center* algorithm as opposed to the other case where the user picks vertices from the entire original meshes.

Users of group A were able to conclude the task in less than half the time as compared to users of group B. We have also measured (a) the percentage of marker pairs that were erroneous (do not correspond to the semantically similar joints) and (b) the percentage of joints that were not covered by a selected marker pair. For group A (a) was around 10% and (b) was around 15%. For group B (a) was around 35% and (b) was around 30%.

Finally, we have observed that users in group B tended to use an average of about 20 to 25 more points than users in group A.

3.3. Optimization and Mesh Correspondence

The approach to establishing a correspondence between the two meshes is based on [ZB13] which computes the geometry correspondence via three steps, select marker point pairs, deform the two models to obtain two simplified meshes and finally establish the correspondence between them. The two main parts of this process require multivariate optimization. Previous work (e.g. [ZB13]) solved the problem with a sparse Cholesky factorization by keeping fixed one set of parameters of the problem and then updated it in every subsequent step. This linear approach to solving this system might not converge since it depends on optimizing two or more different sets of independent variables. By using non-linear optimization as opposed to previous approaches we can ensure convergence if a local minimum exists.

Therefore we have employed different types of non-linear optimizations techniques to optimize the performance of our method. Some of these techniques are: BFGS, L-BFGS algorithm which approximates the BFGS using a limited amount of computer memory, Conjugate Gradient (CG) with approximate derivative and with exact derivative, L-BFGS combined with gradient descent and finally the Levenberg-Marquardt algorithm. Among these methods

we have chosen the most competent method which is L-BFGS with gradient descent.

By selecting an appropriate (i) set of marker pairs and (ii) an optimization method we can implement efficiently the mesh correspondence algorithm. This is performed in three phases: (i) we align the source and target model based on the marker pairs, (ii) we iteratively minimize an energy function to achieve mesh fairing, the result of this minimization will be two models with simplified mesh morphology, and (iii) establish the correspondence between the original meshes based on the common simplified mesh morphology. Figure 2 illustrates the algorithm pipeline of the mesh matching process which is a fully automated process.

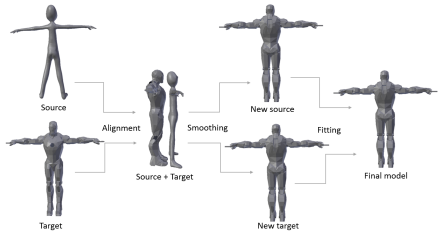

Figure 2: *Step by step mesh correspondence algorithm for a given source and target model.*

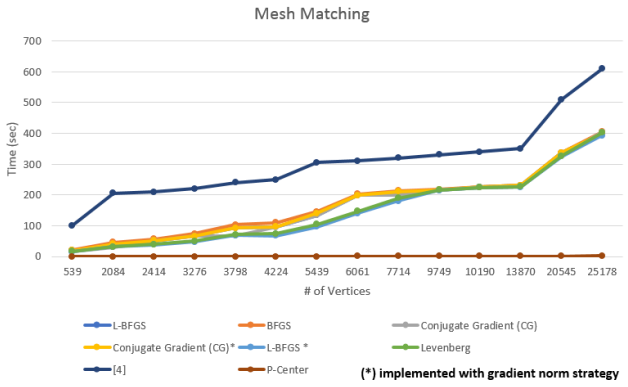

Figure 3: *Mesh correspondence performance evaluation.*

Figure 3 illustrates the performance evaluation of the mesh correspondence process. Original results of the first approach [AGR*16] could not be recomputed since the authors do not provide source code or the details to reproduce their method. We have compared our results with [AGR*16] by using a computationally equivalent platform according to CPU benchmarks. In our experiments we have used a 3.5GHz AMD FX(tm)-8320 which provides similar performance with the Intel Core-i7 2.2GHz used in [AGR*16].

As compared to [AGR*16], our method is almost 40% faster on the average when we use L-BFGS with gradient descent (shown as L-BFGS* in Figure 3) but we have also noticed a slight improvement when using the (CG) method. A key observation is that the most time-consuming procedure seems to be the mesh fairing step.

[AGR*16] evaluates time needed for the selection of manual points (40-60 minutes for each pair). In our case the overall time for this process requires less than 20 minutes for each pair which is consistent with our user study.

Purple line in Figure 3 shows the performance evaluation of the *P-center* algorithm. As compared to [AGR*16] our method provides a better approach for selecting markers. Also the user saves a substantial amount of time by choosing marers from a set of representative points.

4. Skin and Skeleton Transfer

After establishing a correspondence between the geometry (vertices) of the two meshes we perform skeleton and weight transfer. The skeleton and weight transfer is an automated process. Finally, the results should be checked by an artist to ensure that the animated mesh satisfies the requirements and no artifacts are detected during animation.

The first step is the skeleton transfer. Our goal is to create a new skeleton for the target based on the skeleton of the source. As skeleton we refer to a set of joints which represent the motion properties of the object. More specifically every joint affects a set of vertices which are connected with the joint (bone) through a non zero weight in the skinning configuration.

Our approach re-targets each joint of the source based on the corresponding vertices and adapts it to the target by adjusting its orientation and rotation to match the corresponding vertices in the target mesh. After we compute the correct orientation and rotation we can confirm that the target model has the same animation behavior as the source model. Figure 4 illustrates the skeleton transfer pipeline for a given source and target model through an automated process and Figure 5 refers to the final step which is the skinning transfer.

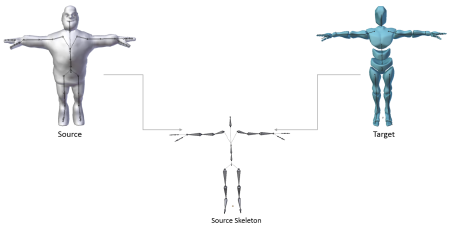

Figure 4: *Work-flow of skeleton transfer.*

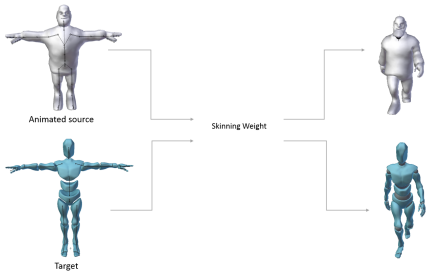

Figure 5: *Work-flow of skin transfer.*

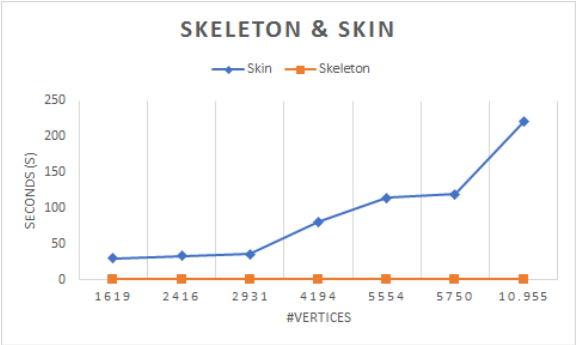

Figure 6: *Skeleton and weight transfer performance evaluation.*

The final step of the animation setup transfer is to re-target the weights. The set of weights among joints and vertices is called skinning and determines how the bone movement will affect the mesh. Transferring correctly all the weights from source to target can derive a correctly animated target model.

Our approach generates a new set of weights for the target from the source by computing new weights based on the mesh correspondence and the skeleton transfer. This is accomplished in two steps. The first step is to generate a new weight for the target vertex based on an existing source vertex by using a simple linear combination. The second step is to generate a target vertex weight without having a source vertex that lies close enough to the target vertex on the simplified common mesh topology, or have more than one vertices that lie close enough on the simplified common mesh topology. In this case we implement a blending method based on the neighborhood of the vertex to eliminate skin deformations such as aliasing and artifacts.

Finally, Figure 6 shows the performance evaluation of the skeleton and weight transfer process. The orange line corresponds to skeleton transfer and it does not depend on the size of the meshes.

5. Discussion and Future Work

We have reported on the development of a user-friendly integrated tool for transferring an animation setup of a source character to a target mesh. This approach enables a user to create new animated characters from scratch in a couple of hours and take advantage of all animations available for the source target.

The entire platform was developed on a 3.5GHz AMD FX(tm)-8320 Eight-Core Processor with 32GB RAM. The models that are supported are Wavefront .obj for static meshes and Collada .dae files for animation setups. All the models that we have used for experiments are from MIXAMO or Free3D which we have adjusted when it was required by using third party software.

For the *P-center* algorithm we have implemented our approach in C++ and have conducted several experiments on many characters with different topologies. The mesh matching algorithm is also implemented in C++. We have used the Dlib optimization library and in house implementations for all the numerical optimization techniques. Skinning and skeleton fitting have been developed in C++.

Model	Vertices	Faces	Time(sec)
Chibi	1.619	2.148	31.06
Iron-Man	2084	3686	32.67
Manikin	2.416	4.624	34.56
Sticky-Boy	4.194	7.592	82.36
Mike	5.554	9.512	115.76
Model	5.750	8.181	120.46
Kratos	10.955	20.862	221.16
Model1	25.178	40.037	546.12

Table 1: *Model details.*

For rendering we have used Opengl 4.5, and for user interaction we have used GLFW.

Table 1 provides the details for these models along with the time for the overall transfer. The third column illustrates the time for skin and skeleton transfer between the same source and different target models. Figure 7 illustrates an example of a full animation transfer between two meshes.

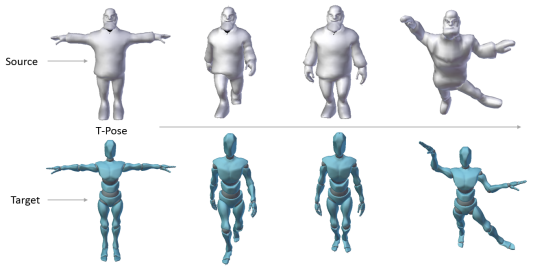

Figure 7: *Experiments of a full animation transfer pipeline in different poses.*

Based on these results we can suggest some future research directions. We will investigate how to make our algorithm fully automated and easy to use by any user. Finally, we would like to combine different source animations to generate a new target animation as a result.

6. Conclusions

We have reported on the development of a user-friendly integrated tool for transferring an animation setup of a source character to a target mesh. This approach enables a user to create new animated characters from scratch in a couple of hours and take advantage of all animations available for the source target.

7. Acknowledgment

This research is co-financed by Greece and the European Union (European Social Fund- ESF) through the Operational Programme "Human Resources Development, Education and Lifelong Learning" in the context of the project "Strengthening Human Resources Research Potential via Doctorate Research" (MIS-5000432), implemented by the State Scholarships Foundation (IKY).

References

[ACP03] ALLEN B., CURLESS B., POPOVIĆ Z.: The Space of Human Body Shapes: Reconstruction and Parameterization from Range Scans. *ACM Trans. Graph. 22*, 3 (July 2003), 587–594. 1

[AGR*16] AVRIL Q., GHAFOURZADEH D., RAMACHANDRAN S., FALLAHDOUST S., RIBET S., DIONNE O., DE LASA M., PAQUETTE E.: Animation Setup Transfer for 3D Characters. *Computer Graphics Forum 35*, 2 (2016), 115–126. 1, 2, 3

[AHLG*13] ALI-HAMADI D., LIU T., GILLES B., KAVAN L., FAURE F., PALOMBI O., CANI M.-P.: Anatomy Transfer. *ACM Trans. Graph. 32*, 6 (Nov. 2013), 188:1–188:8. 1

[DATTS08] DE AGUIAR E., THEOBALT C., THRUN S., SEIDEL H.-P.: Automatic Conversion of Mesh Animations into Skeleton-based Animations. *Computer Graphics Forum 27* (2008). 2

[Gon85] GONZALEZ T. F.: Clustering to Minimize the Maximum Intercluster Distance. *Theoretical Computer Science 38* (1985), 293 – 306. doi:https://doi.org/10.1016/0304-3975(85)90224-5. 2

[LD12] LE B. H., DENG Z.: Smooth Skinning Decomposition with Rigid Bones. *ACM Trans. Graph. 31*, 6 (Nov. 2012), 199:1–199:10. 2

[SLT*19] SIAROHIN A., LATHUILIERE S., TULYAKOV S., RICCI E., SEBE N.: Animating Arbitrary Objects via Deep Motion Transfer. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (June 2019). 1

[vKZHCO11] VAN KAICK O., ZHANG H., HAMARNEH G., COHEN-OR D.: A Survey on Shape Correspondence. *Computer Graphics Forum 30*, 6 (2011), 1681–1707. 1

[WL08] WAREHAM R., LASENBY J.: Bone Glow: An Improved Method for the Assignment of Weights for Mesh Deformation. In *Proceedings of the 5th International Conference on Articulated Motion and Deformable Objects* (Berlin, Heidelberg, 2008), AMDO '08, Springer-Verlag, pp. 63–71. 2

[ZB13] ZELL E., BOTSCH M.: ElastiFace: Matching and Blending Textured Faces. In *Proceedings of the Symposium on Non-Photorealistic Animation and Rendering* (New York, NY, USA, 2013), NPAR '13, ACM, pp. 15–24. 1, 2

[ZHS*05] ZHOU K., HUANG J., SNYDER J., LIU X., BAO H., GUO B., SHUM H.-Y.: Large Mesh Deformation Using the Volumetric Graph Laplacian. In *ACM SIGGRAPH 2005 Papers* (New York, NY, USA, 2005), SIGGRAPH '05, ACM, pp. 496–503. 1