AutoSkin: Skeleton-based Human Skinning with Deep Neural Networks

Xuming Ouyang\textsuperscript{1}, Cunguang Feng\textsuperscript{2}

\textsuperscript{1}School of Software, Shandong University, Jinan, Shandong, 250101, China
\textsuperscript{2}School of Information Science and Technology, Peking University, Beijing, 100871, China

\textsuperscript{3}Corresponding author’s e-mail: oyxm@mail.sdu.edu.cn

Abstract. Articulated character animation has received increasing attention in applications like game or film production. One of the main challenges is the task of skinning that animator should specify the association between skeleton structure and skin surface which involves large amounts of manual weight painting and deformation tuning. In this paper, we propose a deep learning based framework for skinning. By using our framework, the skinning is seen as a multi-regression task and the network can infer the skeleton-skin association and compute the accurate weights automatically while only requiring minimal manual tuning. One significant advantage of our framework is that the impressive performance can be also achieved while dealing with complicated 3D characters, even these characters exist over a hundred bones or dress delicately. Our experiments have indicated that our framework could generate competitive results compared with the commercial software and we aim to apply it in the industry such as games or video applications.

1. Introduction
As is well-known, animating a 3D character plays a significant role in computer graphics, especially in video games and film applications. So far, skeleton-driven animation\cite{1-2} is the most common technique of character animation. By using this method, skeletons are firstly embedded into the 3D character and attached to mesh on the surface. This process is often referred to as rigging. After that, animators need to skin the articulated character that determines the association between skeleton joints and skin surfaces.

A main challenging step of skinning is the weight painting which assigns appropriate bone weights to vertices based on bone-vertex influences. Although there are already excellent performances in weight painting and a variety of 3D applications such as 3DS Max, most of them need to calculate and set the weights manually. This means it is extremely calculation expensive and time-consuming so that it becomes a demanding in character animation. To solve this problem, Li et al.\cite{3} assigns bone weights to vertices based on Projection Distance, but this approach is limited to special types of animated characters.

In our paper, we propose a new algorithm that designs the deep neural networks to compute weights for character skinning automatically. One significant strength of DNNs is that they can learn the probability distribution from a large dataset and extract low- level and high-level features of data. Our goal is to make use of such a strength for character skinning and weight selection. It is worth mentioning that this algorithm is the first method that uses deep learning techniques for articulated character
skinning. In this case, we would like to bridge the gap between articulated character animation and deep learning techniques. To be more precise, we hope to use deep neural networks to learn how to compute and paint weights automatically to a single mesh vertex on the skin surface related to the skeleton. In this paper, we number bones in sequence so that each one of the skeletons has a unique index. Then, we build a vector with the length equal to the number of bones for each mesh vertex, to describe how bones influence this mesh. That means the element in vector represents the weight of corresponding bone with the same index to this mesh vertex. Based on previous experience, we find that each mesh vertex is skinned with up to four bones. That means there is only a maximal four non-zero value in each vector. Therefore, once we can find which four or fewer elements in the vector is non-zero and how large each non-zero value is, the automatic weight painting can be achieved easily.

2. Approach Overview
In our approach, the skinning task is treated as a multi-regression problem. Using data acquired from the skinned character database which is introduced in Section3, DNNs are trained and the weight selection of each mesh vertex on the skin surface can be represented by the neurons in the DNNs. Given this representation, a mapping is generated between the articulated character and the weight. Therefore, the weights of mesh vertices can be estimated automatically when input data propagates forward through DNNs. Once the weights have been determined and painted onto the 3D articulated character, the deformation of this character can be produced properly. Our Networks are composed of multiple stages and thus the estimated result can be refined iteratively via sequential propagation as described in Section4. Using our framework, users can determine the association between skeleton and skin mesh automatically and ensure the high-quality mesh deformation.

3. Data Acquisition
To enhance the performance, the training data is initially extracted from 3D articulated characters before feeding into the model. For every vertex on the skin surface, the data is composed of two different parts that each part describes its unique information and can represent some significant features. In this section, we would like to illustrate them concretely.

3.1. Position information
Position information is the most important feature in our algorithm. As mentioned above, the 3D character is initialized to a standard posture and centred in the global coordinate. Therefore, the global position of vertices on the skin surface can be represented by the 3D coordinates \( v_i = (x_i^V, y_i^V, z_i^V) \), \( i = 1, \ldots, M \) where \( M \) is the number of vertices. Similarly, the global position of bones can be considered as the 3D coordinates of their geometric centre \( b_j = (x_j^B, y_j^B, z_j^B) \), \( j = 1, \ldots, N \), \( N \) means the 3D-character has \( N \) bones. Then the relative position between vertex \( i \) and the bone \( j \) can be considered as a vector

\[
r_{ij} = v_i - b_j = (x_{ij}, y_{ij}, z_{ij})
\]

\[
x_{ij} = x_i^V - x_j^B, \quad y_{ij} = y_i^V - y_j^B, \quad z_{ij} = z_i^V - z_j^B
\]

Likewise, we can calculate the matrix \( r_i \) which has the dimension \( 3 \times N \) to depict the position of a vertex with respect to the whole skeleton

\[
r_i = \left( r_{i1}^T, r_{i2}^T, \ldots, r_{iN}^T \right) \quad \mathfrak{f} = , \ldots, N
\]

With the relative coordinates \( r_i \), the vertex position including distance and direction to every bone is easily obtained so that we can determine which bone this vertex belongs to, and how long the distance between it and other bones. In general, this information is closely related to weight painting and can illustrate the position association between skeleton and skin vertices.
3.2. Normal vector information

Normal vector information is extracted to describe the spatial association between the skeleton and skin vertices which cannot be presented by position information. To explain the skeleton-skin correlation more precisely, the normal vector \( \mathbf{m}_i = (x_i^N, y_i^N, z_i^N, 1) \) should be computed at first. At the next step, we should find the so-called pedal vector that begins from a vertex and ends to the intersection of vertex normal vector and the perpendicular from a bone. For each vertex \( i \), we build a matrix \( \mathbf{p}_i \) with dimension \( 3 \times N \), the \( j \)-th column \( \mathbf{p}_{ij} \) expresses the pedal vector between the normal vector \( i \) and perpendiculars from bone \( j \) which can be calculated by Equation (3).

\[
\mathbf{p}_{ij} = \mathbf{m}_i \cdot \mathbf{u}_{ij} \\
\mathbf{p}_i = (\mathbf{m}_i^T, \mathbf{u}_{i1}^T, \ldots, \mathbf{u}_{iN}^T) \quad j = 1, \ldots, N
\]

The normal vector information is intuitively helpful. For instance, we find the relative position between hands and legs is so close that the vertices of the hand and leg surface have extremely similar position information. However, the normal vector between these two parts of vertices has the opposite direction. Hence, combined with the normal vector information, the spatial associations between skeleton and skin surface can be depicted more comprehensive.

To sum up, we extract the position and the normal vector information from 3D characters. As the input feature of a vertex, this two information should be concatenated to a \( 6 \times N \) matrix \( \mathbf{f}_i \) which is shown in Equation (5).

4. Network Architecture

In our approach, we train the model by using multi-stage networks. Furthermore, the training in each stage consists of three phases that play different roles respectively. In this section, we will introduce the architecture of our networks in detail.

4.1. Multi-stage Networks

Design of the model is inspired by Cao at al. [3] and Wei at al. [4] and the architecture is shown in Figure 1. The predictions are iteratively refined over sequential stages, \( t \in \{1, \ldots, T\} \). At each stage, the learning process will be supervised by intermediate information to avoid the gradient vanishing. After the features propagate through the first stage, the network will estimate a prediction that contains the weight of a vertex. This prediction is a \( N \)-elementary vector that the element \( j \in \{1, \ldots, N\} \) represents the influence weight of bone \( j \) to this skin vertex. In successive stages, a new \( 7 \times N \) dimensional feature matrix that concatenated by the prediction from the previous stage and original features will be fed into the network and used to achieve refinement of prediction.

\[
\mathbf{w}_i^1 = S^1(\mathbf{f}_i) \\
\mathbf{w}_i^t = S^t(\mathbf{w}_{i,t-1}^t, \mathbf{f}_i) \quad t = 1, \ldots, T
\]

As a multi-regression task, the L2-distance between the ground truths \( \mathbf{w}^* \) and the predicted weights \( \mathbf{w}^t \) is minimized for optimization to update the network parameters. A regularization term is added for the non-zero value in the target weight vector \( \mathbf{w}_i^* \) in order to emphasis the importance of the non-zero weight since a single mesh vertex is influenced by 4 bones at most so that there is maximal 4 non-zero weight in each weight vector. We guide the iterative prediction by applying the loss functions at the end of each stage. Specifically, our loss function of a 3D character at each stage is shown in Equation (8) and our final objective is the weighted sum of the loss functions of all stages is shown in Equation (9).
4.2. Multi-phase Networks

In each stage, our network architecture can be composed of three special purposed phases. As an example, we introduce the architecture of the first stage in detail. In the first phase of the stage, the input information is fed into two convolutional layers for analyzing the position and normal vector information of a vertex to a bone. The first layer will extract features from the position information and the normal vector information separately by kernels with size $3 \times 1$ and stride $3 \times 1$. Next, the second layer will use a kernel with a size $2 \times 1$ and stride $1 \times 1$ to conclude these two information together and produce the spatial features which should be inputted to the second phase. The second phase is implemented by a 3-layer CNN to extract the correlative features of a vertex with different bones. For the whole 3 layers, we use the $1 \times 3$ kernels with different strides, respectively. During the last phase, two fully connected layers are carried out to perform a non-linear transformation of the feature maps.

Compared with the first stage, there is only one difference in other stages. Due to the intermediate supervision, the shape of the inputs is $7 \times N$, then in the second layer in phase 1, we should use the $3 \times 1$ kernels instead of $2 \times 1$ kernels to conclude the position, normal vector information as well as the prediction from the previous stage together as we illustrated in Figure 1.

5. Experimental Results

Some cases are shown in Figure 2. For each character, we select some screenshots to present our results in different views. The skeleton and skinning weights are computed automatically by using our AutoSkin framework. The characters have diverse number of mesh vertices, shape as well as costume. In the top row characters are presented in initial pose and the following are our results in different viewpoints or poses.

\[
L' = \frac{1}{M} \sum_{i=1}^{M} \| w' - w^* \|_2^2, \quad \forall = \cdot, \cdots, T
\]

\[
L = \sum_{i=1}^{T} \alpha^i \cdot L'
\]
5.1. Implementation
There are 350 skinned articulated characters in our dataset and each character consist of 243 bones including the skeleton of character as well as the frame joints of costume. The number of mesh vertices on the skin surface is between 10000 and 15000 depending on the size and costume of characters. We split the dataset into the training set (270 characters) to learn the representation, the validation set (50 characters) to select the optimal network and the test set (30 characters) to generalize the network. The number of training epoch and learning rate are set to 300 and 0.0001 respectively. The loss function we use is the weighted mean square error with \( \lambda = 3.5 \). The network training is implemented on a single NVIDIA GeForce GTX 1080 and the total training time is around 1 day.

5.2. Metrics
We evaluate the performance in two different metrics - position and value accuracy. On the one hand, since a single mesh vertex is influenced by maximal 4 bones, the position accuracy is used to check whether our framework can help mesh vertices find the bones which have influence on them; on the other hand, after we found the correct bones, we also need to assess the weights of these bones. Therefore, the value accuracy is utilized.

5.3. Experimental result
Table1 describes the performance of the network with respect to the metrics. The result coincides with our expectation mentioned in section3, it’s clearly that comparing with only using the position information, make use of the position and normal vector information could lead to a better performance measured in both position accuracy and value accuracy.

We also assess the performance of different network architecture and the details are shown in Table2. It is obviously that the accuracies of the CNN are much higher than the fully connected network and RNN. In addition, the position accuracy and value accuracy increase dramatically from 0.911 and 0.894 to 0.932 and 0.913 respectively when we extend the CNN to 2 stages. While expanding the network to 3 stages, the accuracies continue to go up slightly. In this case, the performance does not improve obviously.

Table 1. Evaluation of using different data. The model performs best when both position and normal vector information are used to describe the skeleton-skin association.

| Data Type | Position Accuracy | Value Accuracy |
|-----------|-------------------|----------------|
|            |                   |                |

5
Position & Normal Vector | 93.2% | 91.4%
---|---|---
Only Position | 91.0% | 88.4%
Only Normal Vector | 74.6% | 61.9%

Table 2. Evaluation in different stages. The performance is better when we use CNN rather than RNN or fully connected network. Besides, expanding the CNN block will improves performance significantly. When increasing to three blocks, the result continues to be refined but slightly.

| Base Model   | Position Accuracy | Value Accuracy |
|--------------|-------------------|----------------|
| CNN-1stage   | 91.1%             | 89.4%          |
| CNN-2stage   | 93.2%             | 91.3%          |
| CNN-3stage   | 93.9%             | 32.0%          |
| RNN          | 89.6%             | 87.4%          |
| Fully Connected | 90.2%         | 87.6%          |

5.4. Comparison with BonesPro

BonesPro is a sophisticated, fast and optimized technology for organic skinning of characters and objects in Autodesk 3ds Max. In-game or film production, artists usually animate characters initially with some tools such as BonesPro, and then fix them manually. Figure 3 presents the result comparison of BonesPro, ours and ground truth. We select two typical characters and show the front and back views of each one. With our framework, there are few misassociated that animators can refine them easily and effectively by hand. On contrast, BonesPro might lead to wrong skinning results in a large area such as the skirt and the accessories. To fix all such areas, animators must be experienced and spend large amounts of time.

6. Conclusion

In this paper, we proposed a deep learning-based framework for the skinning of articulated character animation. This framework allows us to treat the skinning procedure as a multi-regression task so that the skeleton-skin binding could be determined automatically and their weights are also computed at the same time. Compared with other state-of-the-art automatic skinning techniques, our method can deal with more complex character models, even with a complicated costume or hundreds of bones.

Acknowledgments

The authors are supported by the National Key R&D Program of China 2018YFB1004100.

References

[1] Magnenat-Thalmann, N., Laperrire, R., & Thalmann, D. (1988). Joint-dependent local deformations for hand animation and object grasping. In Proceedings on Graphics interface’88.
[2] Kavan, L., Collins, S., Žára, J., & O'Sullivan, C. (2007, April). Skinning with dual quaternions. In Proceedings of the 2007 symposium on Interactive 3D graphics and games (pp. 39-46). ACM.
[3] Li, J., Lin, F., Liu, X., & Wang, H. (2018). High Performance Automatic Character Skinning Based on Projection Distance. 3D Research, 9(1), 9.
[4] Cao, Z., Simon, T., Wei, S. E., & Sheikh, Y. (2017). Realtime multi-person 2d pose estimation using part affinity fields. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 7291-7299).
[5] Wei, S. E., Ramakrishna, V., Kanade, T., & Sheikh, Y. (2016). Convolutional pose machines. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 4724-4732).