SkinningNet: Two-Stream Graph Convolutional Neural Network for Skinning Prediction of Synthetic Characters

Albert Mosella-Montoro and Javier Ruiz-Hidalgo
Universitat Politècnica de Catalunya
{albert.mosella, j.ruiz}@upc.edu
https://imatge-upc.github.io/skinningnet

Abstract

This work presents SkinningNet, an end-to-end Two-Stream Graph Neural Network architecture that computes skinning weights from an input mesh and its associated skeleton, without making any assumptions on shape class and structure of the provided mesh. Whereas previous methods pre-compute handcrafted features that relate the mesh and the skeleton or assume a fixed topology of the skeleton, the proposed method extracts this information in an end-to-end learnable fashion by jointly learning the best relationship between mesh vertices and skeleton joints. The proposed method exploits the benefits of the novel Multi-Aggregator Graph Convolution that combines the results of different aggregators during the summarizing step of the Message-Passing scheme, helping the operation to generalize for unseen topologies. Experimental results demonstrate the effectiveness of the contributions of our novel architecture, with SkinningNet outperforming current state-of-the-art alternatives.

1. Introduction

Animating a 3D character is a complex and time-consuming process that animators spend years learning to do efficiently. In a typical animation pipeline, an artist first creates a mesh model and specifies the skeleton topology, skinning weights are then painted manually. During this process, the animators follow two main steps. First, they do the skin binding, which consists of defining which parts of the mesh will be affected by the movement of a specific joint. Finally, they decide the skinning weights that describe which amount of movement is transferred to the skin.

SkinningNet is a Two-Stream Graph Convolutional Network that, given an input mesh and its corresponding skeleton, performs the skin binding and then computes the skinning weights for each mesh vertex as it can be seen in Fig. 1. Previous methods rely on pre-computed handcrafted features to create the mesh and skeleton relationship. The proposed method automatically extracts the best features to relate the mesh and the skeleton to predict the associated skinning weights. The main contributions of this paper are: a) A Two-Stream Graph Neural architecture that learns to extract features from meshes and skeletons with different topologies, b) A Multi-aggregator Graph Convolution (MAGC) layer that extends the Message-Passing scheme to use a multiple aggregation approach that better generalizes for unseen graph topologies, c) A novel skin binding method that uses a skeleton joint representation instead of a bone representation and d) A Mesh-Skeleton Graph Convolution Network that exploits this skeleton joint representation to find the optimal relationship between the input mesh and the skeleton. All these contributions allow the proposed method to outperform current state-of-the-art with over 20% improvement on mesh deformation error. Furthermore, experimental results show the ability of the proposed method to generalize for complex mesh and skeleton structures of different domains.

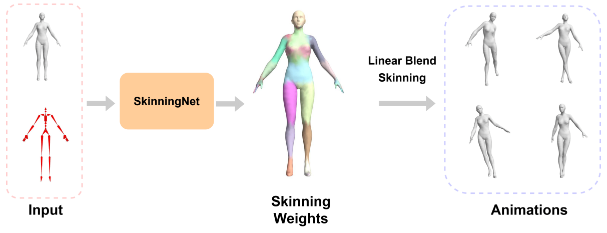

Figure 1. **Overview** of the proposed pipeline. Given an input mesh and its associated skeleton, SkinningNet predicts the skinning weights used by the Linear Blend Skinning [26] algorithm to create a set of animations.

2. Related Work

This section reviews techniques related to Graph Neural Network solutions that are used for Geometric Learning and then transitions to more specific techniques used for skinning weight prediction.

2.1. Graph Neural Networks

Graph Neural Networks can be divided into two subsets: the ones that generalize the convolution operation using a spectral approach [3, 6] and the ones that use a spatial approach [9, 17, 27]. The vast majority of the new proposed graph convolutions can be seen as an adaptation of the Message-Passing scheme [11]. The Graph Convolution Network (GCN) [17] is one of the popular message-passing implementations that proposes to transform the nodes of a neighbourhood using a weight matrix that is normalized by the degree of the neighbourhood. The transformed node features are aggregated using the addition operator.

Another widely adopted graph convolution is Graph Attention Network (GAT) [34] that proposes learning an attention coefficient using edge attributes, which are defined as the difference between the central node and its neighbours. Edge Convolution Network [35] is one of the implementations widely adopted on 3D Geometric tasks, which consists of defining the edge attributes of a graph using an asymmetric function. Edge attributes of each neighbourhood are fed into a shared Multi-Layer Perceptron (MLP) and aggregated using maximum or addition aggregators. Residual connections [5, 21, 28] have also been shown to help achieve better performance on deep Graph Neural Networks. Recently, Multi-Neighbourhood Graph Convolutions [29] introduced the combination of multiple neighbourhoods to create an enriched node descriptor.

Most of the previously described works use mean, maximum and addition aggregators, which fail to distinguish between neighbourhoods with identical features but different cardinalities, as proved by Xu *et al*. [38]. To solve that, Dehmamy *et al*. [7] proposed to use multiple aggregators. We propose an extension of the multi-aggregator scheme using complementary aggregators and learning how to combine them instead of just concatenating or adding the results of each of them. The motivation for the use of multiple scalers is to improve the generalization of the convolution for unseen topologies while avoiding the value of each aggregation exploding when the degree of the neighbourhood increases. In SkinningNet, this extension allows the network to generalize for more complex and unseen mesh and skeleton topologies.

2.2. Skinning Weight Prediction

The techniques used to compute the skin deformation of synthetic characters are usually needed by real-time applications, such as video-games. Approaches such as Linear Blend Skinning (LBS) [26] or Dual Quaternion Skinning (DQS) [14] are widely used due to their simplicity and computational efficiency. These techniques compute the deformation of the mesh based on a set of skinning weights that are assigned to each of the mesh vertices. The skinning weights are either painted manually by an animator or automatically generated. Automatic skinning weight prediction techniques can be grouped into two different categories: geometric based methods [1, 8, 12, 14, 15, 18, 19, 30, 33, 36] and data-driven solutions [13, 16, 20, 25, 31].

Geometric based methods rely on geometric characteristics between meshes and skeletons. The earliest methods to automatically generate skinning weights proposed to exploit Heat Diffusion [2] and Illumination [37] models. Alternatively, other methods used energy functions for the estimation, such as Elastic Energy [15] or Laplacian Energy [12]. Later, Dionne *et al*. [8] proposed Geodesic Voxel Binding to handle non-watertight meshes. All these methods rely on functions that assign the skinning weights depending on the distance between joints and vertices. However, this assumption does not work in practice for AAA game characters that commonly have complex topologies where multiple independent components can intersect.

Data-driven methods typically require multiple poses of a mesh or different meshes as input to learn how to compute the skinning weights. New methods such as [4, 22, 32] estimate skinning weights from Motion Capture data. They focus on finding skinning weights of humanoids and assume a fixed skeleton topology, making the network unable to generalize for characters with different skeletons.

NeuroSkinning [24] is one of the earliest proposed methods to automatically compute the skinning weights for synthetic characters using neural networks. This method makes use of graph convolutions to compute the skinning weights of a new mesh. NeuroSkinning uses a bone representation, meaning that the network predicts the skinning weights per bone instead of per joint. It also relies on creating a super-skeleton that consists of the fusion of all the skeletons that can be found in the training set. This super-skeleton is needed to cope with the fixed output of the network. As a result, this assumption makes the network unsuitable for working with skeletons that can not fit in the super-skeleton structure.

RigNet [39] tries to overcome this limitation using a k-NN approach. The network predicts the skinning weights only for the k-nearest bones to the mesh vertex. This feature allows the network to work with unseen skeleton topologies, however, the bone representation is still used. By definition, the movement of a skeleton driven mesh comes from the rotation of a joint. In an ideal scenario where each of the joints has one associated bone, both representations are equivalent. However, this assumption does not work for complex meshes, where a joint has more than one associated bone. The bone representation used in RigNet has problems to manage these scenarios, common in stylized characters. To overcome this problem, our proposal directly uses a skeleton joint representation that can manage com-

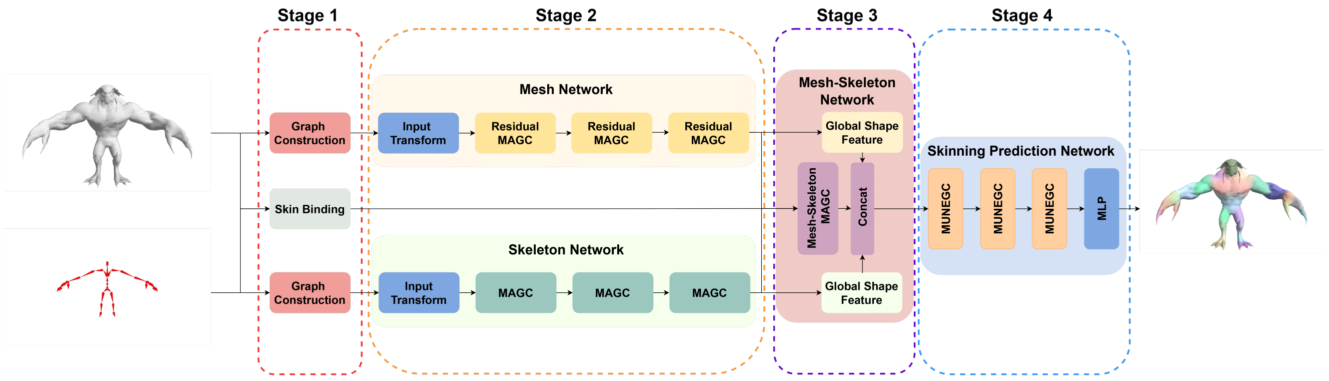

Figure 2. **SkinningNet** architecture is composed of four main stages. Stage 1 is in charge of building the needed graphs from the input mesh and its associated skeleton. Stage 2 is responsible for extracting features independently for the mesh and skeleton. Stage 3 combines the previous mesh and skeleton features to extract a descriptor that relates both structures. Stage 4 predicts the skinning weights.

plex skeleton topologies where each of the joints can have more than one bone. Furthermore, both RigNet and Neuroskinning rely on handcrafted features to learn the relation between a mesh and its associated skeleton. Our proposal consists of a Two-Stream Graph Neural Network that learns the relation between mesh and skeleton on training time, selecting the best set of features automatically without having to rely on selected handcrafted features.

3. SkinningNet

This section details the proposed SkinningNet architecture. Sec. 3.1 gives a general overview of the four stages of the proposed architecture. Sec. 3.2 explains how neighbourhoods are selected in the graph's creation step from the input meshes and their associated skeletons. Sec. 3.3 formulates a novel graph convolution layer, *Multi-Aggregator Graph Convolution* (MAGC), where multiple aggregators are used to allow the network to distinguish between neighbourhoods with similar features but different cardinality. Finally, Sec. 3.4 explains how the different Graph Convolutional Blocks in SkinningNet are constructed based on the proposed MAGC.

3.1. Architecture Overview

SkinningNet is a Two-Stream Graph Convolutional Neural Network, that takes as input a mesh and its corresponding skeleton and predicts a set of skinning weights, one for each mesh vertex. It is composed of four different stages as depicted in Fig. 2.

Stage 1: *Graph Construction* and the *Skin Binding* (explained in detail in Sec. 3.2). The Graph Construction step converts the mesh and the skeleton inputs into two independent graphs. The Skin Binding step decides which joints will influence each vertex and creates a graph representing this relationship. The Graph Construction output is fed into Stage 2 and the output of Skin Binding is used in Stage 3.

Stage 2: *Mesh and Skeleton Networks*. These networks transform the node attributes to feature vectors through an input transform implemented using an MLP. Each network is responsible for extracting features independently for the mesh and skeleton. The *Mesh Network* is composed of three Residual MAGC layers, whereas the *Skeleton Network* is composed of three MAGC layers, further details are given in Sec. 3.3 and 3.4. This difference is mainly because the skeleton is usually much simpler than the mesh and does not require a deep network to learn the characteristics of its geometry. The output of both networks will be combined in the Stage 3.

Stage 3: *Mesh-Skeleton Network*, based on a single Mesh-Skeleton MAGC layer. This block relates the mesh and the skeleton using the output of the *Skin Binding* step. The output of this block is a single graph where each node can represent the vertices of the mesh or the joints of the skeleton. However, only the nodes representing the vertices of the mesh are used on the following stages. Furthermore, to help with the final skinning weight prediction in Stage 4, a global shape descriptor that encodes the global information of the mesh and skeleton graphs is extracted and concatenated to the mesh nodes.

Stage 4: *Skinning Prediction Network*. The final vertex skinning weight is predicted for each vertex of the mesh. It is composed of three Multi-Neighbourhood Graph Convolution (MUNEGC) layers followed by an MLP. Here, the MAGC is exploited in a multi-neighbourhood fashion, combining the mesh topology and the local shape information to extract an enriched descriptor used to predict the skinning weights. Further details are given in Sec. 3.4.

3.2. Neighbourhood Construction

Identifying the neighbours of a node is an important step of Graph Convolutional Networks, as connections between nodes (edges) act as the receptive field on conventional CNNs. SkinningNet uses different strategies to create neighbourhoods, specific to each type of graph: i.e. *mesh*, *skeleton* and *mesh-skeleton* graphs. In the mesh graph, faces are converted into undirected edges. Additionally, a radius-based neighbourhood is created over the mesh structure, where the k-random nodes inside of a radius r are connected to the central point of the neighbourhood. The radius-based neighbourhood is used by the MUNEGC operation. In the skeleton graph, the bones are converted to undirected edges. Finally, the relation created by the *Skin Binding* step is used to decide the connections in the mesh-skeleton graph.

The *Skin Binding* step is in charge of assigning which of the joints are going to influence each of the vertices. As stated in the related work section, previous works [24, 39] have based the skin binding process on a bone representation, where each vertex has a set of bones assigned. This assignation is done using a k-NN approach that associates the k nearest bones to each of the vertices. In this work, the proposed *Skin Binding* is based on a joint representation that is more natural than the bone representation. It uses the closest bones to select which are going to be the joints that influence each of the vertices. In particular, for each vertex of the mesh, the closest bones of the skeleton are found and the associated root joint for each bone is selected. Finally, that selection is refined leaving only the k unique joints. The entire algorithm is described in Algorithm 1.

Algorithm 1: k-unique joints of the closest bones

```
Input: Vertex positions, joints position and bones
Output: The selected joints for each vertex
foreach $vertex$ do
    Compute distance $d(vertex, bones)$
    Sort distances $d$
    Replace bones with their associated root joint
    Select the k unique joints
end foreach
```

3.3. Multi-Aggregator Graph Convolution (MAGC)

The Multi-Aggregator Graph Convolution (MAGC) is an extension of the Message-Passing scheme [11], where multiple aggregators are used to let the graph convolution layer distinguish between neighbourhoods with identical features but with different cardinalities. The workflow of the MAGC is depicted in Fig. 3. The first step is to compute the messages that each of the neighbours are sending to the central node of the neighbourhood. These messages are a function based on the attributes of each edge $E_{ji} = \mathcal{F}(X_j, X_i)$ where X_j and X_i denote the features of nodes j and i respectively. The messages are combined using different aggregators. The aggregators proposed in this work are formally defined in Eq. 1 where $N(i)$ represents the neighbourhood of the node i.

$$A = \begin{cases} A^{max} = \underset{j\in N(i)}{max}(E_{ji}) \\ A^{min} = \underset{j\in N(i)}{min}(E_{ji}) \\ A^{mean} = \underset{j\in N(i)}{mean}(E_{ji}) \\ A^{std} = \underset{j\in N(i)}{std}(E_{ji}) \end{cases} \tag{1}$$

The results of each aggregator are scaled using a set of logarithmic degree scalers. The proposed scalers are: *i) identity*, the value of the aggregator is not changed; *ii) amplification*, the value of the aggregator is amplified, and *iii) attenuation*, the value of each aggregator is attenuated. Eq. 2 formalizes the proposed scalers where d_{train} refers to the mean degree of the whole training split. The motivation for the use of different logarithmic scalers is to improve the generalization of the convolution for unseen topologies, avoiding that the value of each aggregation explodes when the degree of the neighbourhood increases.

$$S = \begin{cases} S^{amp} = \dfrac{log(d)}{log(d_{train})} \\ S^{att} = \dfrac{log(d_{train})}{log(d)} \\ S^{iden} = 1 \end{cases} \tag{2}$$

Finally, the resulting operation of applying each scaler to each aggregator is fed into an MLP that learns how to fuse the information. Eq. 3 defines the combination of aggregations and scalers, with A being the set of aggregation operations and S the set of scaler operations. The combination of two sets of operations is described as $\otimes$ so $\mathcal{M}$ is the combination of all the scalers with all the aggregators.

$$\left.\begin{aligned} A = \{A^{max}, A^{min}, A^{mean}, A^{std}\} \\ S = \{S^{iden}, S^{amp}, S^{att}\} \\ \mathcal{M} = S \otimes A \end{aligned}\right\} \tag{3}$$

The resulting MAGC layer is described in Eq. 4, where all combinations $\mathcal{M}$ are fused using an MLP network to produce the output feature of node i and layer l in a feed-forward neural network.

$$X_i^l = \text{MLP}\left(\underset{j\in N(i)}{\mathcal{M}}(E_{ji})\right) \tag{4}$$

3.4. Graph Convolutional Blocks

The proposed architecture extends the previously defined MAGC, producing three types of *Graph Convolutional Blocks*: Residual MAGC, Mesh-Skeleton MAGC and Multi-Neighbourhood Graph Convolution (MUNEGC).

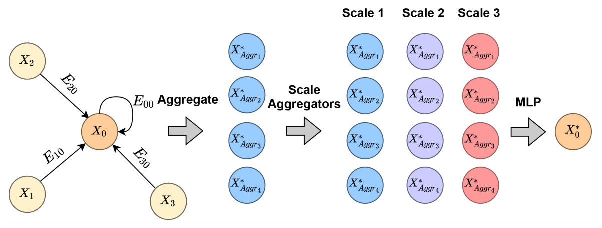

Figure 3. **Multi-Aggregator Graph Convolution** workflow. Where E_{ji} is the message sent from node j to node i.

The **Residual MAGC** is based on the ideas proposed in [5, 21, 28]. Each Residual MAGC is composed of two MAGCs stacked together with a short connection. In the case that the input and the output have different dimensionalities, the short connection contains a function $\mathcal{P}$ that projects the input feature space to the output ones. The Residual MAGC is formalized in Eq. 5 where $\{\text{MAGC}_s\}$ represents s stacked MAGC layers inside the correspondent block.

$$X_i^l = \{\text{MAGC}_s\} + \mathcal{P}\left(X_i^{l-1}\right) \tag{5}$$

The **Mesh-Skeleton Graph Convolution** employs a modification of MAGC to deal with graphs where the neighbourhood is composed of an heterogeneous combination of nodes. To combine the mesh and the skeleton nodes, a one-hot vector is concatenated to the node feature. In this way, features corresponding to the mesh node can be distinguished from features corresponding to skeleton nodes by the graph operation. Eq. 6 shows the one-hot concatenation proposed for the combination:

$$X_i^l = \begin{cases} \text{concat}\left\{[0], X_i^l\right\} & \text{if} \quad i \in \text{mesh} \\ \text{concat}\left\{[1], X_i^l\right\} & \text{if} \quad i \in \text{skeleton} \end{cases} \tag{6}$$

The **Multi-Neighbourhood Graph Convolution** is an extension of MAGC based on [29] where two kinds of neighbourhoods are used to get the new node feature. Eq. 7 defines the extension of MAGC to a multi-neighbourhood approach. Where $\mathcal{K}$ is the set of neighbourhood types:

$$X_i^l = \text{MLP}\left(\underset{k\in\mathcal{K}}{\text{concat}}\left\{\text{MLP}\left(\underset{(j)\in N^k(i)}{\mathcal{M}}\left(E_{ji}^k\right)\right)\right\}\right) \tag{7}$$

4. Results

4.1. Dataset

The proposed method has been trained and evaluated on the **RigNetv1** [39] dataset which is publicly accessible for non-commercial use. This dataset is composed of 2703 rigged characters of different categories. The original split of the dataset is followed where 2163 assets are used for training, 270 for validation and 270 for testing. All training assets contain between 1k and 5k vertices, and all of them are scaled between $[-1, 1]$ and oriented to face the same direction. The number of skeleton joints is in a range from 3 to 48 with a mean of 25. In addition, two more assets from the **Paragon Collection** [10] have been used for the generalization study in Sec. 4.8. These two assets have been simplified so the resulting meshes contain about 6k vertices and 50 joints and normalized as done in **RigNetv1**. The **Paragon Collection** dataset is allowed to be used in non-interactive linear media products under a non-exclusive and non-transferable license.

4.2. Implementation Details

The detailed architecture with the number of filters used in each layer is shown in Table 1. Several attributes are selected as initial features for nodes in the mesh and skeleton structures to help the network to learn the relation between both structures. The node attributes used to describe the mesh are the normalized 3D coordinates of the vertices, the Geodesic distance to the $k = 5$ unique joints from the nearest bones, the start and end positions of the bone of those joints and a boolean value indicating if the joint is an end joint or not. In the case of the skeleton, only the normalized 3D position of the joints is included. Sec. 4.6 discusses the advantages and disadvantages of using the Geodesic distance with respect to the Euclidean. To compute the Geodesic distance between joints and vertices, the volumetric version proposed in [39] is employed, where the shortest path from vertex to joint passing through the interior mesh volume is computed.

The radius for the second neighbourhood used in the Skinning Prediction Network is $r = 0.06$. From all nodes inside the radius r, a maximum of 10 nodes are randomly sampled to create the neighbourhood. The final output of SkinningNet is the skinning weights for the $k = 5$ unique joints of the closest bones of each vertex. A dropout layer is added before each Fully Connected layer of the Skinning Prediction Network with a probability of $p = 0.5$. The network is trained in an end-to-end fashion for 200 epochs. The Rectified Adam (RAdam) [23] optimizer is used with a learning rate of 1×10^{-4}, a weight decay of 1×10^{-4}, and a batch size of 4. The Kullback-Leibler divergence loss is used to minimize the distance between the predicted skinning weight distribution and the ground truth distribution.

4.3. Comparison With The State-of-the-art

SkinningNet is compared to the two most recent data-driven approaches: NeuroSkinning [24] and RigNet [39]. Both networks have been trained from scratch following the procedure described in their respective papers. In the case of NeuroSkinning, the original Euclidean distance has been replaced with the Geodesic one, as it will be demonstrated in Sec. 4.6 to be a better choice for watertight meshes. Furthermore, to guarantee a fair comparison, the same input at-

Mesh Network	
Layer	**N.Filters**
Input Transform	MLP(64, 128)
Residual MAGC	128
Residual MAGC	256
Residual MAGC	512
Skeleton Network	
Input Transform	MLP(64)
MAGC	128
MAGC	256
MAGC	512
Mesh - Skeleton Network	
Mesh Global Shape	MLP(256)
Skeleton Global Shape	MLP(256)
Mesh-Skel MAGC	512
Concat	512 + 256 + 256
Skinning Network	
MAGC	256
MAGC	128
MAGC	64
MLP	(64, 32, k)

Table 1. **SkinningNet** architecture details with the number of filters used in each layer. k is the number of joints that can influence each of the vertices.

tributes used in SkinningNet and described in Sec. 4.2 have been used in the two architectures. Both methods proposed a deep learning architecture based on graph convolutional layers. In the case of NeuroSkinning the Graph Attention layer [34] is used, whereas RigNet uses the Edge Convolution layer [35]. Four different metrics have been used to evaluate the skinning weight prediction:
1. *Precision and Recall* finds the set of joints that influence each vertex significantly, where influence corresponds to a prediction larger than 1×10^{-4}, as described in [24, 39].
2. *L1-norm* of the difference between the predicted skinning weight and the ground truth vectors of each vertex of the mesh. The average of this metric is computed across the whole test split.
3. *Deformation error* computes the Euclidean distance between the position of the vertices deformed after applying the predicted skinning weights and the ground truth ones. To compute this metric, 10 different random poses are computed where all the joints in the skeleton are randomly rotated within a range of ± 10 degrees.

SkinningNet outperforms the best method of the state-of-the-art with over 5% of improvement on Precision with the same Recall and 15% improvement on average L1-norm. Table 2 summarizes the comparison with the state-of-the-art. Fig. 4 shows the skinning weight prediction examples on three assets. A random color is assigned to each skeleton joint and colors are blended between vertices in the mesh using the skinning weights. It can be seen, for instance in the *bat* asset in the middle row, how the proposed algorithm better predicts skinning weights with associated colors that are closer to those of the ground truth in the first column.

Method	**Prec(%)**	**Rec.(%)**	**avg L1**
NeuroSkinning [24]	82.3	79.7	0.41
Rignet [39]	82.3	**80.8**	0.39
SkinningNet	**87.0**	**80.8**	**0.33**

Table 2. **Prediction results** comparison with the current state-of-the-art techniques.

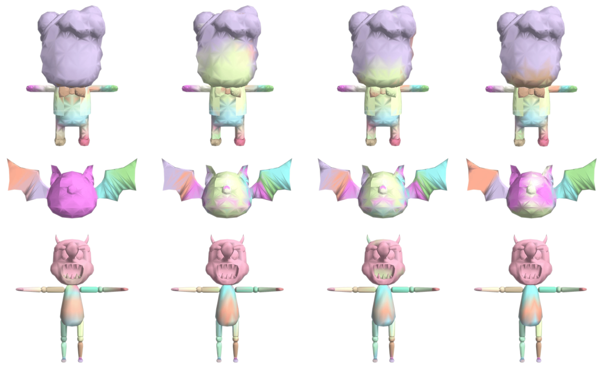

Figure 4. **Skinning weights** prediction results of each of the state-of-the-art methods. A random color is assigned to each joint and colors are blended between vertices in the mesh using the skinning weights.

Even though results are significantly better than state-of-the-art, the Precision, Recall and L1-norm metrics are not good enough to evaluate skinning weight predictions. The reason is that, in the case of Precision and Recall, the magnitude of the skinning weight is not taken into account. Furthermore, in the case of the L1-norm, it does not completely capture the importance of an error on the skinning weights. Two similar L1 errors can result on different deformation errors depending on whether the L1 error comes from small differences on each of the associated joints or it is concentrated on a single joint. For these reasons, the deformation error will be used for the rest of the experiments. Table 3 shows the average and the maximum deformation error. Again, SkinningNet outperforms with a 20% of improvement in the average error and with a 17% of improvement in the maximum error. The qualitative results of this metric can be observed in Fig. 5, where SkinningNet is able to generate reasonable results where previous state-of-the-art methods fail.

4.4. Architecture Design Study

In this section the influence of each of the components of the network is studied. Table 4 shows the results of remov-

Method	Avg. Def	Max. Def
NeuroSkinning [24]	0.002843	0.2151
Rignet [39]	0.002921	0.2246
SkinningNet	**0.002288**	**0.1789**

Table 3. **Deformation error** comparison with the current state-of-the-art techniques. The errors are inside a range of $[0, 0.2]$ and are calculated over the normalized version of each mesh.

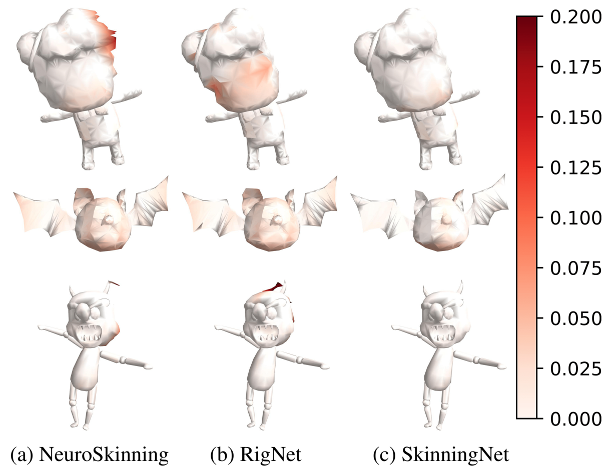

Figure 5. **Deformation error** of three different characters with a randomly generated pose.

ing the global shape feature, the residual connections and the MUNEGC from the original SkinningNet.

Method	Avg. Def	Max. Def
Baseline	**0.002288**	**0.1789**
No Global Feature	0.002452	0.1905
No Residual	0.002394	0.1862
No MUNEGC	0.002427	0.2009

Table 4. **Architecture Study** that shows the influence of each of the proposed stages in the output.

Removing the MUNEGC layers from the network leads to a loss of performance of about 5%. These results show that MUNEGC is helping to have an enriched local descriptor for each of the vertices. This descriptor enables the network to be aware of the local structure around the vertices. Similar performance losses are obtained when removing the *Global Shape Feature* as it helps the network to be aware of the global shape of the skeleton and the mesh when doing the prediction. Finally, adding the residual connections to the *Mesh Network* leads to an improvement of 4%, demonstrating that the residual connections are helping the proposed approach.

4.5. Joint vs. Bone Representation Study

In this section, the difference between a joint vs. a bone representation is analyzed. The *Skin Binding* step is modified to use bones instead of joints when creating the relations between the mesh and the skeleton. Table 5 shows the results of both approaches. As can be seen, the skeleton joint representation gives an improvement of 5% in both average and maximum deformation. This improvement is due to the fact that the joint representation follows a natural approach where each of the joints represents an articulation which is in charge of defining the movement of each bone.

Method	Avg. Def	Max. Def
Joint	**0.002288**	**0.1789**
Bone	0.002407	0.1893

Table 5. **Joint vs. Bone** representation study, where the influence of each representation is analyzed.

4.6. Euclidean vs. Geodesic Distance Study

Finding the vertex to joint distance is a critical step for skinning prediction methods. NeuroSkinning [24] proposed the use of the Euclidean distance while RigNet [39] proposed to use the Geodesic distance. Both distances have their advantages and disadvantages. The Geodesic distance is defined for connected components, with the distance between two non-connected components being infinity. This means that the Geodesic distance is better suited for watertight meshes, whereas the Euclidean distance can be used for both watertight and non-watertight meshes. In terms of performance, using Geodesic distance helps the network to predict better results than using Euclidean distance as observed in Table 6.

Method	Avg. Def	Max. Def
Euclidean	0.002663	0.4473
Geodesic	**0.002288**	**0.1789**

Table 6. **Geodesic vs. Euclidean** performance comparison.

An example of the effects of a higher maximum error when using the Euclidean distance can be observed in Fig. 6 where the tail of the squirrel is deformed with respect to the Geodesic result.

4.7. Graph Convolution Study

In this work, the use of the Multi-Aggregator Graph Convolution (MAGC) is proposed. To study the influence of this operator on the output of the prediction, the MAGC has been replaced by three different state-of-the-art graph convolutions. Table 7 details the results of this study, showing

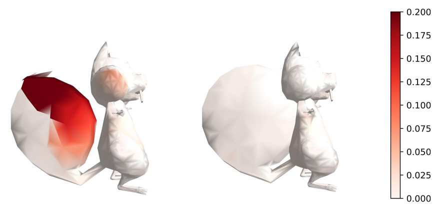
(a) Euclidean (b) Geodesic

Figure 6. **Euclidean vs. Geodesic** qualitative results. It can be observed that the euclidean distance introduces big deformation errors in the tail.

that Edge Convolution [35] achieves the closest results to the proposed MAGC. The improvement in respect to the Edge Convolution is about 4% on the average deformation and about 6% on the max deformation, demonstrating that the MAGC helps to predict better skinning weights.

Method	Avg. Def	Max. Def
MAGC	**0.002288**	**0.1789**
EdgeConv [35]	0.002381	0.1921
GAT [34]	0.002551	0.2098
GCN [17]	0.002765	0.2533

Table 7. **Graph Convolution** study where MAGC has been compared with three different state-of-the-art operators.

4.8. Generalization Study

A generalization study was performed to evaluate if the proposed method has good generalization and is suitable for working with AAA game characters. The trained networks using RigNetv1 [39] have been applied to two high quality assets from the Paragon Collection [10].

Method	Avg. Def	Max. Def
NeuroSkinning	0.003724	0.1213
Rignet	0.003398	0.1051
SkinningNet	**0.002666**	**0.0664**

Table 8. **Generalization Study** of the state-of-the-art methods. The deformation error is computed using a normalized version of the Paragon Assets [10].

Results in Table 8 show that SkinningNet outperforms previous works with over 28% improvement in average deformation and 36% in maximum deformation error, proving that the proposed method generalizes better than previous methods for unseen complex characters. In Fig. 7, it can be observed that the proposed method is generating good quality animations without strong errors, whereas the other methods have high errors in both characters.

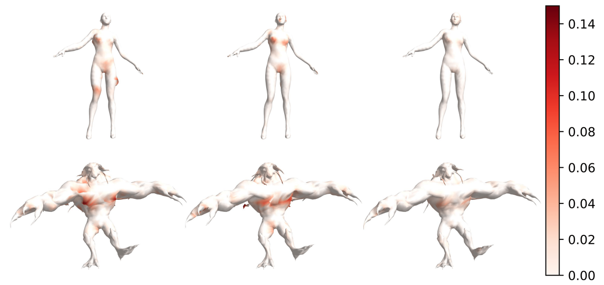
(a) NeuroSkinning (b) RigNet (c) SkinningNet

Figure 7. **Generalization study** with Aurora and Rampage assets from Paragon collection [10].

5. Conclusions and Limitations

This work presents SkinningNet, a Two-Stream Graph Convolutional Neural Network that automatically generates skinning weights for an input mesh and its associated skeleton. The SkinningNet architecture is based on the novel Multi-Aggregator Graph Convolution layer that allows the network to better generalize for unseen topologies. Moreover, the proposed joint-based skin binding and Mesh-Skeleton network learns to find the best relationships between the mesh and skeleton helping to improve the final skinning weight predictions. The proposed architecture outperforms current approaches with over a 20% improvement on mesh deformation error and is also able to better generalize for complex characters of unseen domains.

Even though results are promising there are some limitations that could be addressed. The first one is that assigning the joints that will influence each of the vertices is based on a k-NN approach. The network can learn which of these joints will affect each vertex, however, if the joint that should influence the vertex is not inside the initial k proposal, the network will not be able to find it. To overcome this issue, we propose to explore link prediction strategies to let the network learn the binding strategy. The second one is that the current configuration uses the standard Geodesic distance that, by definition, is infinity on non-connected regions, meaning that the network could experience a drop on its performance in non-watertight meshes. To overcome this limitation, new approximations to the Geodesic distance using voxelization could be explored. Finally, the use of the deformation error as a loss could be explored to improve the performance of the current approach.

Acknowledgements. This work has been partially supported by the project PID2020-117142GB-I00, funded by MCIN/AEI /10.13039/501100011033. The authors would like to thank Denis Tome for his technical advice during the development of this project.

References

[1] Seungbae Bang and Sung-Hee Lee. Spline interface for intuitive skinning weight editing. *ACM Trans. on Graphics*, 37(5), 2018. 2

[2] Ilya Baran and Jovan Popović. Automatic rigging and animation of 3d characters. *ACM Trans. on Graphics*, 26(3), 2007. 2

[3] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. Spectral Networks and Locally Connected Networks on Graphs. *CoRR*, abs/1312.6, 2013. 2

[4] Xu Chen, Yufeng Zheng, Michael J Black, Otmar Hilliges, and Andreas Geiger. SNARF: Differentiable Forward Skinning for Animating Non-Rigid Neural Implicit Shapes. In *International Conference on Computer Vision (ICCV)*, 2021. 2

[5] Lingwei Dang, Yongwei Nie, Chengjiang Long, Qing Zhang, and Guiqing Li. Msr-gcn: Multi-scale residual graph convolution networks for human motion prediction. In *ICCV*, 2021. 2, 5

[6] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering. In *NIPS*, 2016. 2

[7] Nima Dehmamy, Albert-László Barabási, and Rose Yu. Understanding the representation power of graph neural networks in learning graph topology. In *NIPS*, 2019. 2

[8] O. Dionne and M. de Lasa. Geodesic binding for degenerate character geometry using sparse voxelization. *IEEE Trans. Vis. & Comp. Graphics*, 20(10), 2014. 2

[9] David Duvenaud, Dougal Maclaurin, Jorge Aguilera-Iparraguirre, Rafael Gómez-Bombarelli, Timothy Hirzel, Alán Aspuru-Guzik, and Ryan P. Adams. Convolutional networks on graphs for learning molecular fingerprints. In *NIPS*, 2015. 2

[10] Epic Games. *Paragon Collection*. Epic Games, 2018. `https://www.epicgames.com/`. 5, 8

[11] Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. Neural message passing for quantum chemistry. In *ICML*, 2017. 2, 4

[12] Alec Jacobson, Ilya Baran, Jovan Popoviundefined, and Olga Sorkine. Bounded biharmonic weights for real-time deformation. *ACM Trans. on Graphics*, 30(4), 2011. 2

[13] Doug L. James and Christopher D. Twigg. Skinning mesh animations. *ACM Trans. on Graphics*, 2005. 2

[14] Ladislav Kavan, Steven Collins, Jiří Žára, and Carol O'Sullivan. Skinning with dual quaternions. In *I3D*, page 39–46, 2007. 2

[15] Ladislav Kavan and Olga Sorkine. Elasticity-inspired deformers for character articulation. *ACM Trans. on Graphics*, 31(6), 2012. 2

[16] Meekyoung Kim, Gerard Pons-Moll, Sergi Pujades, Seungbae Bang, Jinwook Kim, Michael J. Black, and Sung-Hee Lee. Data-driven physics for human soft tissue animation. *ACM Trans. on Graphics*, 36(4), 2017. 2

[17] Thomas N Kipf and Max Welling. Semi-Supervised Classification with Graph Convolutional Networks. In *International Conference on Learning Representations (ICLR)*, 2017. 2, 8

[18] Martin Komaritzan and Mario Botsch. Projective skinning. *Proc. ACM Comput. Graph. Interact. Tech.*, 1(1), 2018. 2

[19] Martin Komaritzan and Mario Botsch. Fast projective skinning. In *Proc. MIG*, 2019. 2

[20] Binh Huy Le and Zhigang Deng. Robust and accurate skeletal rigging from mesh sequences. *ACM Trans. on Graphics*, 33(4), 2014. 2

[21] Guohao Li, Matthias Müller, Ali Thabet, and Bernard Ghanem. Deepgcns: Can gcns go as deep as cnns? In *ICCV*, 2019. 2, 5

[22] Peizhuo Li, Kfir Aberman, Rana Hanocka, Libin Liu, Olga Sorkine-Hornung, and Baoquan Chen. Learning Skeletal Articulations with Neural Blend Shapes. *TOG*, 40, 2021. 2

[23] Liyuan Liu, Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Jiawei Han. On the variance of the adaptive learning rate and beyond. In *ICLR*, 2020. 5

[24] Lijuan Liu, Youyi Zheng, Di Tang, Yi Yuan, Changjie Fan, and Kun Zhou. Neuroskinning: Automatic skin binding for production characters with deep graph networks. *ACM Trans. on Graphics*, 2019. 2, 4, 5, 6, 7

[25] Matthew Loper, Naureen Mahmood, Javier Romero, Gerard Pons-Moll, and Michael J. Black. SMPL: A skinned multi-person linear model. *ACM Trans. on Graphics*, 34(6), 2015. 2

[26] N. Magnenat-Thalmann, R. Laperrière, and D. Thalmann. Joint-dependent local deformations for hand animation and object grasping. In *Proc. Graphics Interface '88*, 1988. 1, 2

[27] Federico Monti, Davide Boscaini, Jonathan Masci, Emanuele Rodolà, Jan Svoboda, and Michael M Bronstein. Geometric deep learning on graphs and manifolds using mixture model CNNs. In *Proceedings IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017. 2

[28] A Mosella-Montoro and J Ruiz-Hidalgo. Residual Attention Graph Convolutional Network for Geometric 3D Scene Classification. In *ICCVW*, 2019. 2, 5

[29] Albert Mosella-Montoro and Javier Ruiz-Hidalgo. 2d–3d geometric fusion network using multi-neighbourhood graph convolution for rgb-d indoor scene classification. *Information Fusion*, 76:46–54, 2021. 2, 5

[30] Tomohiko Mukai and Shigeru Kuriyama. Efficient dynamic skinning with low-rank helper bone controllers. *ACM Trans. on Graphics*, 35(4), 2016. 2

[31] Yi-Ling Qiao, Lin Gao, Yu-Kun Lai, and Shihong Xia. Learning bidirectional lstm networks for synthesizing 3d mesh animation sequences. *arXiv:1810.02042*, 2018. 2

[32] Shunsuke Saito, Jinlong Yang, Qianli Ma, and Michael J. Black. SCANimate: Weakly supervised learning of skinned clothed avatar networks. In *Proceedings IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, June 2021. 2

[33] Weiguang Si, Sung-Hee Lee, Eftychios Sifakis, and Demetri Terzopoulos. Realistic biomechanical simulation and control of human swimming. *ACM Trans. on Graphics*, 34(1), 2015. 2

[34] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph Attention Networks. *ICLR*, 2018. 2, 6, 8

[35] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E Sarma, Michael M Bronstein, and Justin M Solomon. Dynamic Graph CNN for Learning on Point Clouds. *TOG*, 2019. 2, 6, 8

[36] Rich Wareham and Joan Lasenby. Bone glow: An improved method for the assignment of weights for mesh deformation. In *Proc. the 5th International Conference on Articulated Motion and Deformable Objects*, 2008. 2

[37] Rich Wareham and Joan Lasenby. Bone glow: An improved method for the assignment of weights for mesh deformation. In *Proc. the 5th International Conference on Articulated Motion and Deformable Objects*, 2008. 2

[38] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826*, 2018. 2

[39] Zhan Xu, Yang Zhou, Evangelos Kalogerakis, Chris Landreth, and Karan Singh. RigNet: Neural Rigging for Articulated Characters. *TOG*, 39, 2020. 2, 4, 5, 6, 7, 8