Cloth and Skin Deformation with a Triangle Mesh Based Convolutional Neural Network, Supplementary

Nuttapong Chentanez\textsuperscript{1,2}, Miles Macklin\textsuperscript{1,3}, Matthias Müller\textsuperscript{1}, Stefan Jeschke\textsuperscript{1} and Tae-Yong Kim\textsuperscript{1}

\textsuperscript{1}NVIDIA
\textsuperscript{2}Department of Computer Engineering, Faculty of Engineering, Chulalongkorn University
\textsuperscript{3}University of Copenhagen

We modify the code of MeshCNN [HHF\textsuperscript{19}] from https://github.com/ranahanocka/MeshCNN and Spiral Net++ [GCBZ19] https://github.com/sw-gong/spiralnet_plus to compare with our work. The details will be discussed in this section.

The encoder-decoder variation of MeshCNN [HHF\textsuperscript{19}] takes input and produces output on edges, which is not directly compatible with our problem. Therefore, we preprend a first layer that produces edge features from vertex features. Say an edge’s endpoints are vertices $p$ and $q$ with features $f_p$ and $f_q$, the edge features are computed as $(f_p + f_q, |f_p - f_q|)$. We also append a last layer that produces vertex features by averaging from the surrounding edges. We also use $L_1$ as the loss function. For the pants mesh with 12064 edges, we use the pool sizes of 1800, 1350, 780. From a preliminary test, the last pool size can’t be smaller than 780, as otherwise, the code sometimes reports that it can’t collapse edges and crashes.

For the encoder-decoder variation of MeshCNN [HHF\textsuperscript{19}] we modify the code to use our dataset and output the displacements that minimize the $L_1$ error. We set the number of channels of convolution layers to $[c, 2c, 4c, 8c]$ where $c$ is the smallest value such that

$$U_{\text{MeshCNN}}(d) \geq M,$$

where $U_{\text{MeshCNN}}(d)$ is the number of learnable weights of the MeshCNN network when the number of channels of the convolution layers are set as above and $M$ is the number of learnable weights of our network that the MeshCNN network is compared against. For each case, we run experiments with the number of residual blocks 1, 2, and 3, each with the number of channels computed as above, and report the lowest training and testing errors. We train for 200 epochs and the other parameters are left as default. We cannot however, run the code on the tank-female mesh with 43308 edges as the program ran out of memory on our 16GB GPU, no matter what parameters we tried for the pool sizes, even when the batch size is 1.

For Spiral Net++ [GCBZ19], we modify the code to use our dataset and output the displacements that minimize the $L_1$ error. We set the number of channels of the convolution layers to be $[c, c, c, 2c]$ where $c$ is smallest value such that

$$U_{\text{SpiralNet++}}(d) \geq M,$$

where $U_{\text{SpiralNet++}}(d)$ is the number of learnable weights of the SpiralNet++ network when the number of channels of the convolution layers are set as above and $M$ is the number of learnable

1 Alternative Convolution adapted from [HHF\textsuperscript{19}]

Prior to the convolution operator presented in the paper, we experimented with a convolution operator inspired by the edge based convolution from [HHF\textsuperscript{19}]. It does not perform as well as the convolution operators we proposed in this work but we include it here for reference.

For each edge, compute 4 order invariant features based on its positional neighbors.

2 Details about MeshCNN and Spiral++ experiments

We compare MeshCNN [HHF\textsuperscript{19}] and Spiral++ [GCBZ19] with our network for the Pose to Cloth problems. Our network is an encoder-decoder architecture of the form $D^cD^{2c}D^{4c}D_3D^{4c}$ for various values of $c$. The result is shown in Table 1.

\begin{figure}[h]
\centering
\includegraphics[width=0.5\textwidth]{neighbors.png}
\caption{Neighbors for convolution from vertices onto edge pq. Left) Internal edge. Right) Boundary edge.}
\end{figure}

1. For each edge, compute 4 order invariant features based on its positional neighbors and produce an intermediate tensor of size $c_{in} \times E \times 4$.

2. Do a 2D convolution, where the kernel size is $1 \times 4$, no padding, stride 1, with $c_{in}$ input channels and $c_{out}$ output channels, to produce an intermediate tensor of size $c_{out} \times E \times 1$, for which we drop the last dimension of the tensor.

3. For each vertex, aggregate the features from the surrounding edges by averaging.
weights of our network that the SpiralNet++ network is compared against.

As in their default choice, the number of channels only doubled at the innermost layer. For each case, we run experiments with the number of latent channels of 16, 64, 256, 1024 and report the lowest train and test errors. We also need to reduce the learning rate to $10^{-4}$ as the default choice of $10^{-3}$ diverges in some cases. We also need to reduce the batch size to 8 for the cases when $x = 80$, as the default choice of 32 runs out of memory. We then run the training for 200 epochs and other parameters are left at their default values.

References


Table 1: Errors of various convolutions for the upsampling for the pants and tank-female mesh for various networks. Our proposed ring based (RC) and ellipse based (EC) convolutions (RC) for various filter length, specified by the subscript are shown. We use the encoder-decoder architecture $D_1^x D_2^y D_4^z D_5^w$ for various $x$. Error is specified as training / testing average L1 error per vertex per degree of freedom $\times 10^{-3}$. We compare against previous work MeshCNN(MeshCN) [HHF 19] and SpiralNet++(Spiral++) [GCBZ19] where we choose the network parameters so that the number of learnable weights are as close as possible but greater than ours. We also include the errors when using our network but with the convolution replaced by alternatives. V4 refers to the vertex convolution adapted from [HHF 19] as stated in the appendix, SP and SPD are the spiral convolution [BBF 19, GCBZ19]. The cells highlighted in green/yellow show the configurations with lowest training/testing errors for each $x$ and filter size.

Table 2: Errors for various architectures of the encoder-decoder network with skip connection using RC13, EC13, SP13 and SPD13 convolutions with either average or max pooling for the tank-female mesh. Error is specified as training / testing L1 error $\times 10^{-3}$. The cells highlighted in green/yellow show the convolution with lowest training/testing errors for each architecture.

Table 3: Errors when training a single network $D_1^x D_2^y D_4^z D_5^w$ with RCmax with data from 1 to 10 meshes. As expected, the error tends to increase as the number of meshes used for training grows, but they still remain relatively low. The result shows that our network architecture allows for the possibility of upsampling multiple meshes with a single network.
Table 4: Errors for various architectures of decoder network using RC$_{13}$, EC$_{13}$, SP$_{13}$ and SPD$_{13}$ convolutions with either average or max pooling for the dress2 mesh for the pose to cloth problem. Error is specified as training/testing L1 error times $10^{-3}$. The cells highlight in green/yellow show the convolution with lowest training/testing errors for each architecture.