A Study on Different Architectures on a 3D Garment Reconstruction Network

by

Jason Klimack

A Thesis in Partial Fulfilment of the Requirements for the Degree of

Master in Artificial Intelligence

From the Faculties
Facultat D'Informàtica de Barcelona (FIB)
Facultat de Matemàtiques (UB)
Escola Tècnica Superior D'Enginyeria (URV)

Universitat Politècnica de Catalunya (UPC) – BarcelonaTech
Universitat de Barcelona (UB)
Universitat Rovira i Virgili (URV)

Supervisor(s):
Meysam Madadi [2], Sergio Escalera[1,2]
[1]Dept. Mathematics and Informatics, Universitat de Barcelona, Catalonia, Spain
[2]Computer Vision Center, Edifici O, Campus UAB, 08193 Bellaterra (Barcelona), Catalonia Spain

Date of Defense
January 27^{th}, 2021

Abstract

In this work, we perform a study of different network architectures for 3D garment reconstruction. In the study, five networks are compared both quantitatively and qualitatively. The dataset used to train each network is CLOTH3D, which consists of six different garment types: tshirt, top, jumpsuit, dress, skirt, and trousers. The baseline architecture is introduced in the CLOTH3D paper [2]. Each of the other four networks are defined by introducing modifications to this baseline: 1) applying central difference convolution (CDC) to GNNs, 2) new pooling based on spectral clustering, 3) applying octave convolution to GNNs, and 4) combining the CDC and pooling networks (CDC-pool). We show that the CDC and pooling networks independently outperform the baseline, octave, and CDC-pool networks quantitatively. As well as, the pooling network is best suited for modelling the complex dynamics present in the dress and skirt garment types, while there is little difference between the networks for the top, tshirt, trousers, and jumpsuit garment types.

Contents

1 Introduction

The simulation of realistic 3D garments has numerous applications involving virtual humans. For instance, this task can be used for creating CGI characters in video games or movies, and can also be applied in the fashion industry by obtaining a virtual fitting of the garment to a human body. In fact, any environment which contains virtual 3D humans can benefit from the use of realistic 3D garments.

In order to provide realism in simulated clothing, the clothes must be properly fitted to a human body, and the dynamics of the clothes must accurately match the expected behaviour based on the pose of the human body. For example, if the human body is bent over forwards, then it is expected that the clothes be tight across the persons back, and loose in the front. If the simulated clothes do not provide these dynamics, then they are not performing realistically.

Learning a 3D garment reconstruction model is a challenging task, mainly due to the lack of available 3D garment data. There are three main approaches for generating 3D garments: 3D scanning, 2D image to 3D model predictions, or by generating synthetic data. First, 3D scanning is a time-consuming task that is prone to missing information. Furthermore, 3D scanned models are in a static state, meaning that each scan will only be able to capture a garment in a single position, which is not useful for simulations involving many human poses. Second, predicting 3D garment data from 2D RGB images is an inaccurate method that cannot capture all of the necessary information required to create the proper dynamics of the garment fabric. Third, generating synthetic data is error-free, and can be easily obtained for multiple body poses and garment types.

Recently, CLOTH3D [2] was introduced as the first large-scale synthetic dataset of 3D clothed human sequences. It consists of over 2 million 3D samples with a large variety of garment types, topologies, shapes, sizes and fabric. The garments were generated on top of thousands of 3D human pose sequences with different body shapes to provide realistic cloth dynamics. Due to the large variability in samples, and realism in the cloth dynamics, CLOTH3D is an excellent source to learn a 3D garment reconstruction model from.

Along with a new dataset of 3D dressed human samples, the CLOTH3D paper also provided a 3D garment reconstruction network based on graph neural networks (GNNs). This network uses a conditional variational auto-encoder (CVAE) to predict the garment shape, such that the clothes provide realistic draping and dynamics over the provided human body. The garments are encoded as offsets from the SMPL [10] body, in a similar fashion as [11]. However, standard SMPL only contains topologies corresponding to a standard body, which is insufficient for encoding offsets for learning skirt-like garments. Thus, the CLOTH3D paper also defines a novel topology that joins the inner legs of the SMPL body, such that the offsets can be properly encoded for learning the skirt and dress garments.

The goal of this work is to achieve a better understanding of GNNs for clothes simulation. This is done by studying different network architectures used for 3D garment reconstruction. Each network is comprised of a modification to the architecture from

CLOTH3D. This architecture is chosen over others for its generalization capabilities of both standard and skirt-like garments. The studies will include the introduction of two convolution operators to GNNs: central difference convolution [15], and octave convolution [3]. Previously, these operators have been applied to convolutional neural networks (CNNs), but have never been applied to a GNN. On top of these convolution operators, we also introduce a new pooling technique designed to increase the networks capability of modelling realistic cloth dynamics. The main idea of this pooling is to use spectral clustering to form groupings of body vertices, and reduce the number of vertices in the final pooling layer to six, such that the resulting vertices correspond to the arms, legs, front and back torso.

The rest of the work is divided into the following sections: section 2 looks at alternative works that have been done on garment generation. Section 3 provides an overview of the network from the CLOTH3D paper. Section 4 describes the alternative architectures implemented in this work, including the original usage of the convolution operators that have been adapted to GNNs. Section 5 describes the experiments performed, including the dataset used for training, training options for each network, the different networks being compared, an ablation study on the capacity of the network, and the final results of the networks. The resulting architectures are compared both quantitatively using mean square error of the garment vertices, and qualitatively using rendered samples. Finally, in section 6, we provide a discussion about the results, and make conclusions based on the experiments performed.

2 Related Work

In this section we describe both past and current works in 3D garment generation. This section begins by describing techniques for garment generation from RGB images and 3D scans, followed by deep learning techniques for fitting garments to human models.

In [16], garment modelling from a single image of a dressed human is performed. First, using human joint locations in the image, the 3D pose and shape of the human is estimated. Second, the outlines of each garment in the image are found. Third, using the estimated human body and projected outlines of the garments, the initial 3D garments are recovered. Fourth, the shading of the input image is used to obtain fine garment details, such as wrinkles. Finally, the fine details are applied to the initial garments, obtaining final 3D garments that well-represent those of the input image. The two main limitations of this approach are: 1) the assumption that the garments are symmetrical in the front and back, in order to model the back side of the garment, and 2) the method does not perform well when there are occlusions in front of the garments.

In [6], a novel framework is introduced for reconstructing high-quality 3D garments with synthesized texture. A dressed human body is scanned using KinectFusion [12], where the garments are then extracted and processed using Instant Field-aligned Meshing [7]. The texture is synthesized separately using DeepTextures [5], and then applied to the 3D garment model.

Deep learning techniques applied to garment reconstruction are demonstrated in [17, 13, 8]. In [8], two main modules are used to generate realistic clothing from 3D scans of dressed humans in motion: 1) a linear subspace model for cloth deformations and factoring out the human body and pose, and 2) a conditional adversarial network is used to generate normal maps of scanned clothes, to add fine details to an otherwise low-resolution garment.

TailorNet [13] was introduced as a model for predicting clothing in 3D as a function of human pose, shape, and garment style. The basis of this model was to capture fine detailed wrinkle information in garments while using a model that can be generalized for any garment type. This was done by separating the problem into high and low frequency components, in order to capture both fine details of the garments, as well as generalizing for any garment type. Each component was predicted independently using a mixture of style-shape specific models and an MLP, for the high and low frequency components, respectively. Finally, the components are added together to obtain the final garment. To this end, we apply the octave convolution [3] to GNNs, which allows the high and low frequency components of the garments to be predicted separately.

In our study, we use the CLOTH3D network as a baseline architecture. This architecture consists of a conditional variational auto-encoder, used to reconstruct a 3D garment to fit a human body. Similar to the work done in [11], the garments are encoded as offsets from the SMPL human body, such that the network can be generalized to predict any garment topology. Additionally, this network utilizes a novel topology for the human body which allows the encoding of skirt and dress garment offsets. More details of this architecture are described in section 3. Counter to the works of [16, 6, 8, 17], we do not use raw images or scanned data to feed the network. Rather, 3D garment data is provided and reconstructed to fit a a given human body.

The work done in [13] showed that the fine-grain garment information can be reconstructed by separating the garment into high and low frequency components. To this end, we apply the octave convolution [3] to GNNs, to observe whether convolving the high and low frequency components of the garments separately provides any additional information. Furthermore, we apply the central difference convolution [15] to GNNs to observe whether the gradient information can be used to learn the fine details of 3D garments.

3 Overview of CLOTH3D Architecture

The CLOTH3D paper introduced two main features: the CLOTH3D dataset of synthetically generated 3D garments, and a deep learning network for 3D garment generation. Due to the large variability in garment types and topologies, creating an architecture that can be generalized for any garment type is difficult. However, a homogeneous dimensionality of the input data can be obtained by encoding the garment vertices as a set of offsets from the body vertices. Thus, the problem is simplified to a prediction of the garment offsets from a given fixed set of body vertices. The data preprocessing techniques are described in section 3.1. Finally, a Graph Conditional Variational Auto-Encoder (GC-

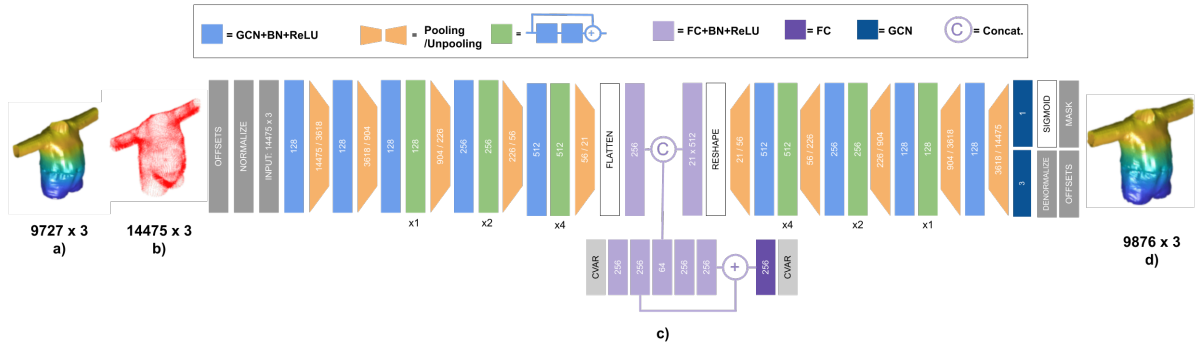

Figure 1: Network architecture from CLOTH3D. a) input garment, b) offsets, c) network architecture, and d) reconstructed garment. Garment images taken from [2].

VAE) is used to predict the garment offsets from a given fixed input topology using convolutional operations, and is discussed in section 3.2.

3.1 Data Preprocessing

A major issue with creating a garment generation network is the vast variation in garment types and topology. In order to get around this issue, the garments can be represented as offsets between the vertices of a fixed SMPL [10] human body topology and the vertices of the garment. Furthermore, a garment mask is used to highlight the body vertices connected to different garment types, which allows different garments to be encoded from all of the body vertices. To obtain the garment offsets from the body, a matching between the garment and body vertices is required for each sequence. Non-rigid ICP [1] is used to register the garment vertices on top of the body vertices. In order to get an accurate matching between the garment and body vertices, both the body and garment are placed in alignment while in the rest pose. Additionally, the vertices of the default SMPL body are not spatially dense enough to get a good matching, thus SMPL was extended to superSMPL by subdividing the body mesh into more vertices. Furthermore, the vertices of the head, hands, and feet are never matched with any garment. Thus, these vertices can be removed from the body, resulting in a lower input dimensionality of 14475 vertices. Finally, the topology of skirts and dresses differs greatly from that of the SMPL body, thus, a novel topology was introduced by connecting the inner leg vertices between the left and right legs, such that the resulting body can be better matched to the skirt/dress topology.

3.2 Network

The goal of the introduced network is to learn a meaningful latent space for garments of any type, shape, or dynamic (wrinkles), which can then be used for generating realistic draped garments. A Conditional Variational Auto-Encoder (CVAE) network is used to predict the latent space of the garments. In the work, separate networks were used for learning the static garments (SVAE) and the dynamic garments (DVAE) which includes the wrinkle information. The VAE network layers are composed of graph convolutions and pooling layers. Conditioning variables, *cvar*, are used in the network to factor out

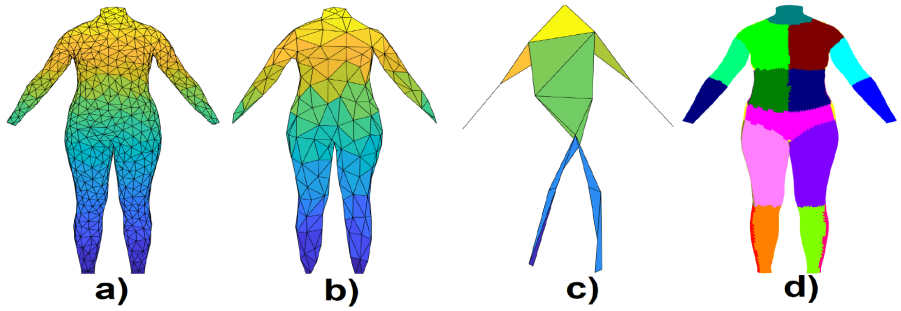

Figure 2: a), b), and c) show steps in the mesh hierarchy for the pooling layers from CLOTH3D. d) shows the original segmented regions. Each point in c) corresponds to a single region in d). Garment images taken from [2].

irrelevant parameters from the latent space. The conditioning variables used in SVAE are the body shape and garment tightness. DVAE on the other hand is conditioned on body shape, garment tightness, body pose, and the latent code from SVAE. The conditioning network is trained separately from the main architecture, and is appended as an additional branch at the end of the encoder. Batch normalization (BN) and ReLU are used for the GCN and FC layers, as the architecture shows in figure 1.

3.2.1 Graph Convolution

The input data for the network is composed of the set of body vertices, as well as the body topology (default or skirt) representing the edge connections between the vertices. Given this input, graph convolutional filters can be used to learn features. The filtering is then computed using spectral graph convolution [4]:

$$y = \sum_{i=0}^{K} w_i T_i(\hat{L}) x \tag{1}$$

where x is the graph node features, w_i are the learnable weights, $\hat{L} = 2L/\lambda_{max} - I$, L is the normalized Laplacian matrix, and $T_i(\hat{L})$ is the i-th Chebyshev polynomial order. The size of the receptive field of each node is K, as defined by Chebysev polynomial order. When $K = 0$, each node only receives information from itself, and no others. By setting $K = 1$, each convolution layer allows each node to receive information from each of its immediate neighbours only. For all of our experiments, we use $K = 1$. Thus, to increase the receptive field of the vertices in the graph, either additional GCN layers are appended to the network, or pooling layers are used, as described next.

3.2.2 Pooling

The pooling operator was implemented by creating a hierarchy of meshes with decreasing number of vertices. There were 6 different levels used with the following number of vertices: $14475 \rightarrow 3618 \rightarrow 904 \rightarrow 226 \rightarrow 56 \rightarrow 21$. An initial segmentation consisting of 21 regions was created from the highest level body mesh (14475 vertices), which was used to restrict the grouping of vertices from different segments. Figure 2 demonstrates the pooling.

3.3 Loss

The conditioning network is trained prior to the VAE, using L1 loss. Its weights are then frozen during the training of the VAE. The loss for VAE is a combination of a garment term, a *cvar* term, and KL-divergence of the latent code. The garment term consists of a summation of the computed L1 losses for the offsets, mesh-face normals, and mask. The *cvar* term is the L1 loss from the conditioning network.

4 Network Changes

The goal of this work is to study different architectures for 3D garment reconstruction. In this section, we introduce three modifications to the architecture presented in section 3. First, a new pooling based on spectral clustering of the vertices in the SMPL body is described in section 4.1. The main objectives of the pooling are 1) obtain six vertices in the final pooling layer representing the arms, legs, front and back torso. 2) force the lower arms and legs to have a higher density of vertices than the rest of the body. This allows for more information to be learned from the arms and legs, where the majority of variability in the garment shape occurs. 3) observe how an alternative grouping of vertices affects the performance of the network. The second modification to the network is the adaptation of the Central Difference Convolution (CDC) operator from CNNs to GNNs. The purpose of CDC is to improve upon the vanilla convolution by incorporating both intensity-level, and gradient-level information. Due to the success that CDC has shown when applied to CNNs, it stands to reason that a similar improvement may occur when CDC is applied to GNNs, as described in section 4.2. The third and final modification to the baseline network architecture is the adaptation of the octave convolution from CNNs to GNNs. The octave convolution is used to separate the input features into high and low frequency components, and perform the convolution operation on each component individually. The main purpose of the octave convolution is to improve upon the efficiency of the vanilla convolution. We apply the octave convolution to GNNs in section 4.3.

4.1 Spectral Pooling

In this work, I create new pooling layers based on spectral clustering [14] of the vertices in the SMPL body. There are four steps involved in this process: 1) split the vertices in two groups corresponding to the front and back of the human body, 2) apply spectral clustering recursively to each group. At each iteration, spectral clustering is applied to each group independently, resulting in $c \times g$ groups, where c is the number of clusters that each group is split into, and g is the number of groups at the current iteration. The process is repeated until only a single vertex remains in each group. 3) Create the final pooling layer by merging groups from the front and back regions, and 4) create the new topology associated with each pooling layer. There are three objectives for this new pooling method. First, is to reduce the graph to six vertices in the final layer, corresponding to the two arms, two legs, front and back components of the main body. Second, is to force the lower arms and legs to have a more dense population of vertices than the remainder of the body. There is higher variability in the clothing for these regions, thus more vertices are required in order for the model to learn this information appropriately. Finally, the third objective is to observe how an alternative grouping method of the vertices affects

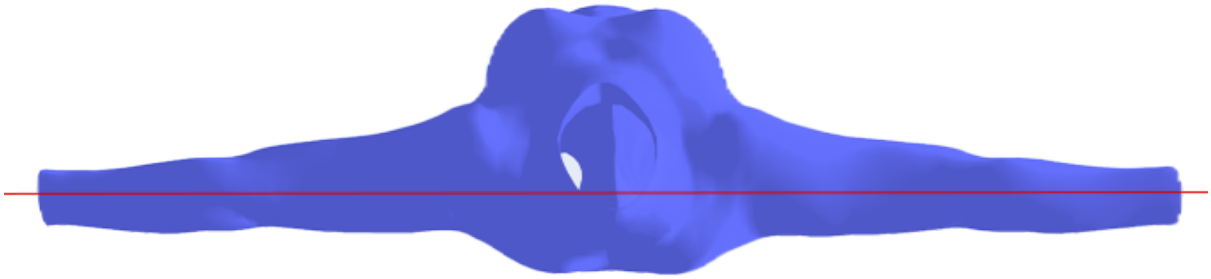

Figure 3: Separation line between the front and back partitions for the initialization of the spectral clustering groupings. There are 8123 vertices in the front partition, and 6352 vertices in the back partition.

the model.

In the first step, I separate the vertices into two groups, corresponding to the front and back of the body. This is done by defining a plane as shown in figure 3. The plane was defined at the approximate midpoint of the arms and legs of the body, so that the main body components (arms, legs, and torso) are preserved in both groups.

Second, I apply spectral clustering to each group independently to achieve 5 clusters each, representing the 5 main body components, and resulting in 10 groups of vertices overall. Next, spectral clustering with 4 clusters is applied recursively to each remaining group until only a single vertex is present in each group. There are 6 resulting pooling layers (therefore 7 levels of vertices by including the initial input) with the following number of vertices in each: $14475 \rightarrow 8648 \rightarrow 2558 \rightarrow 640 \rightarrow 160 \rightarrow 40 \rightarrow 10$.

As stated previously, an objective of the pooling is to obtain six vertices in the final layer, such that each vertex is assigned to a unique limb: two arms, two legs, front and back torso. However, after applying spectral clustering, the final layer consists of 10 vertices. Thus, we define the vertices for the final layer manually by merging the two vertices associated to each limb (arms and legs each contain two) together, therefore reducing 10 vertices to 6.

The fourth and final step is to create the adjacency matrices associated to the vertices at each pooling layer. Provided as input are the new pooling layers, as well as the initial adjacency matrix, A_0 of 14475 vertices. Each new adjacency matrix A_i can be formed by creating an edge between every pair of vertices v_n and v_m in A_i iff there exists an edge between u_j and u_k, where $u_j, u_k \in A_{i-1}$ and $u_j \subset v_n$ and $u_k \subset v_m$. This approach is repeated for both the default and skirt topologies. Figure 4 demonstrates the resulting effects of the pooling.

Due to the increased number of pooling layers, the network architecture has to be modified, while maintaining the current capacity of the network. Two additional pooling-plus-GCN blocks were added to the encoder side of the architecture. In order to maintain the total capacity of the network, the number of GCN layers in each block, as well as the number of features in each layer were modified such that the total number of features in the network is approximately the same as the default network. The layers are then

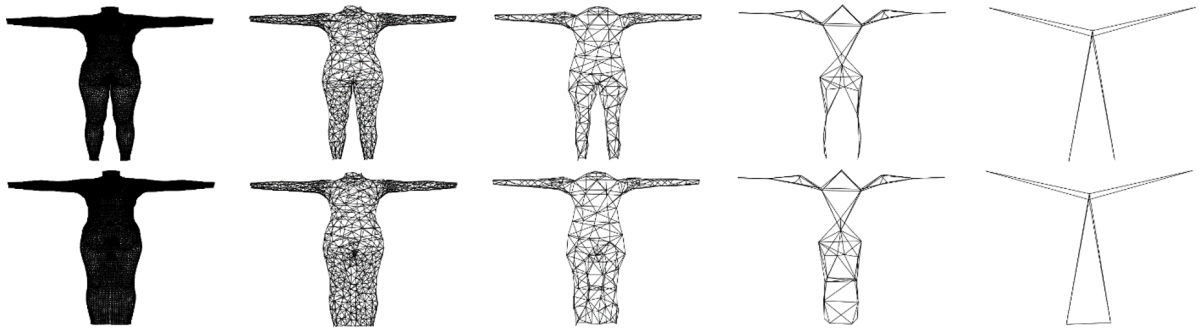

Figure 4: Mesh hierarchy for the new pooling. Top: mesh hierarchy for the standard topology, Bottom: mesh hierarchy for the skirt topology. From left to right, the number of vertices in the mesh are: 14475, 640, 160, 40, 6.

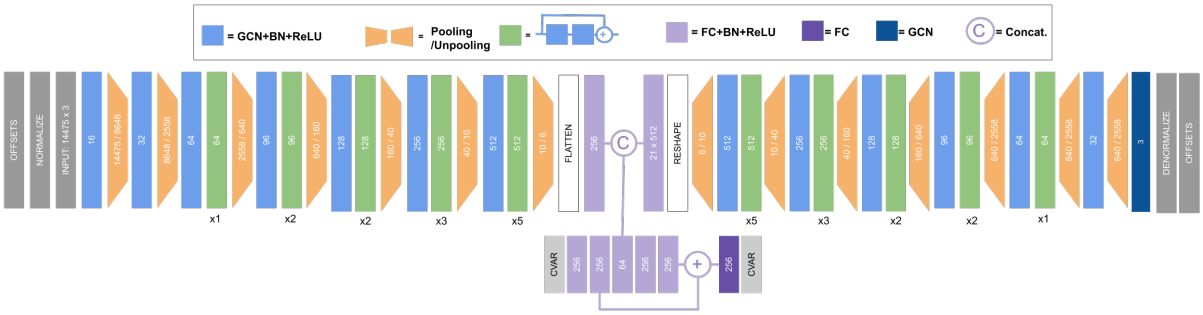

Figure 5: Modified architecture for the additional pooling layers.

mirrored to the decoder side of the architecture. The new architecture is shown in figure 5.

4.2 Central Difference Convolution

The central difference convolution (CDC) operator was introduced in [15] for face anti-spoofing. It was used as a replacement for the traditional convolution operator used on 2D feature maps with additional channel dimension. The goal of CDC is to enhance the generalization capabilities of the vanilla convolution by computing the center-oriented gradient of sampled values. While the aggregation step in the vanilla convolution computes the weighted sum of all the values in each sampled region, the central difference aggregation step computes the weighted sum of the difference between each value in the region and the central value in the region. Finally, in order to achieve a convolution operator that is capable of learning both intensity-level (vanilla convolution) and gradient-level (central difference convolution) information, the results from the vanilla convolution and the central difference convolution are aggregated using a weighted sum. The result of the sum is referred to as the Central Difference Convolution (CDC). The reduced CDC aggregation function can be reduced to the difference from the vanilla convolution to the weighted central difference term:

$$y(p_0) = \sum_{p_n \in R} w(p_n) \cdot (x(p_0 + p_n)) - \theta \cdot x(p_0) \cdot \sum_{p_n \in R} w(p_n) \tag{2}$$

where R is the sampled region around the central pixel p_0, $x(p_0 + p_n)$ is the value of the

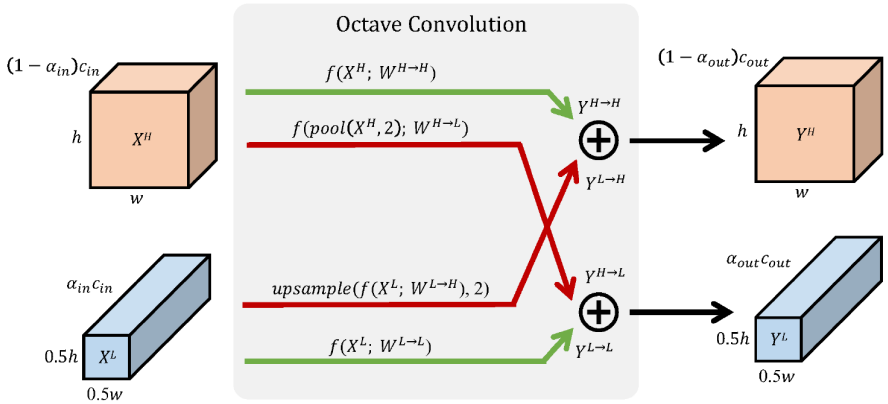

Figure 6: Information exchange between the high and low frequency components of the octave convolution. Green arrows represent the component updating itself, as in vanilla convolution. Red arrows represent information exchange between the two components. Image from [3]. The total number of features before and after the convolution are c_{in} and c_{out}, respectively. h is the height of the input image, w is the width. $f(X; W)$ is the vanilla convolution operator on X with weights W.

sampled pixels, $w(p_n)$ is the weight associated to the position p_n from the center p_0, $x(p_0)$ is the value of the central pixel being sampled, and θ is the weighting factor between the vanilla and central difference operators.

As the CDC was successfully able to achieve an enhanced generalization of the vanilla convolution applied in the 2D domain, it stands to reason that an extension of the CDC to graph convolutions may be able to enhance the generalization of graph (or in this case, garment) information in a similar manner. The graph convolution operator described in section 3.2.1 can be easily adapted to incorporate the CDC, as shown below:

$$y = \sum_{i=0}^{K} w_i T_i(\hat{L})x - \theta \cdot x \cdot w_1 \tag{3}$$

4.3 Octave Convolution

The goal of the octave convolution (OctConv) introduced in [3] is to increase the efficiency of vanilla convolution by separating the feature map into low and high spatial frequency components, and performing the convolution on these components individually. The high and low frequency components differ by an octave [9], with the low frequency component having a lower dimensionality than the high frequency component. Furthermore, the features are split between the high and low frequency components using a ratio α, where there are more features present in the low frequency component. Additionally, the convolution is also designed such that the information update for each component not only occurs from the features of the current component, but also receives information from the opposite component (ie. the low frequency component receives information updates from the high frequency component, and vice versa). Figure 6 depicts the information exchange between the two components.

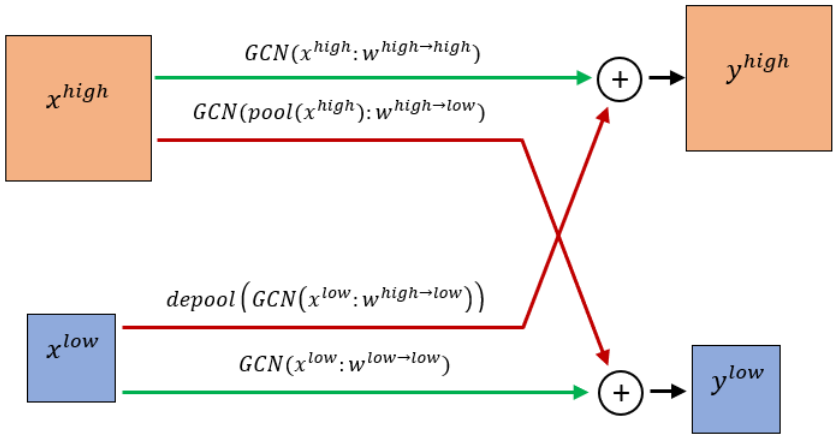

Figure 7: Information exchange of the octave convolution for GNNs (OctGCN). Green lines represent information passing within the same component, and red lines represent information passing between components. For each information exchange, the vanilla graph convolution is applied: $GCN(x : w)$ where x is the input feature matrix, and w is the learnable weight matrix.

The information updates can be implemented efficiently using the vanilla convolution operator with varying inputs (as shown in equation 4). Furthermore, due to the difference in dimensionality between the high and low frequency components, up-sampling and down-sampling must be used to add low frequency data to high frequency, and high frequency data to low frequency, respectively.

$$y^A = \sum_{p_n^A \in R^A} w^{A \to A}(p_n^A) \cdot (x^A(p_0^A + p_n^A)) + G^A(\sum_{p_n^B \in R^B} w^{B \to A}(p_n^B) \cdot (x^B(p_0^B + p_n^B))) \quad (4)$$

where A and B represent the high and low frequency components respectively, if the component to be updated is the high frequency component. Vice-versa otherwise. $G^A(..)$ is the transition function: if A is high, then G^A is upsampling, otherwise $G^A(..)$ is down-sampling.

In order to use OctConv in the garment generation network, there are two steps that need to occur: 1) the OctConv needs to be adapted to handle graph input instead of 2D images, and 2) the encoder/decoder of the network from section 3 needs to be modified to handle the separate high and low frequency components, rather than a single component. The first step is easily implemented using equations 4, but replacing the vanilla convolution operator with the graph convolution from section 3.2.1. Figure 6 shows the exchange of information for the octave graph convolution (OctGCN).

The second step requires a near doubling of effort for the encoder and decoder; mainly, there are now two components that need to be tracked throughout the network, which require two separate instances of pooling at each pooling layer. Finally, in order to maintain a consistent input and output of the overall network, as well as the network bottleneck with that of the network from section 3, the first layer of both the encoder and decoder each receive only the high frequency component as input, and the final layers

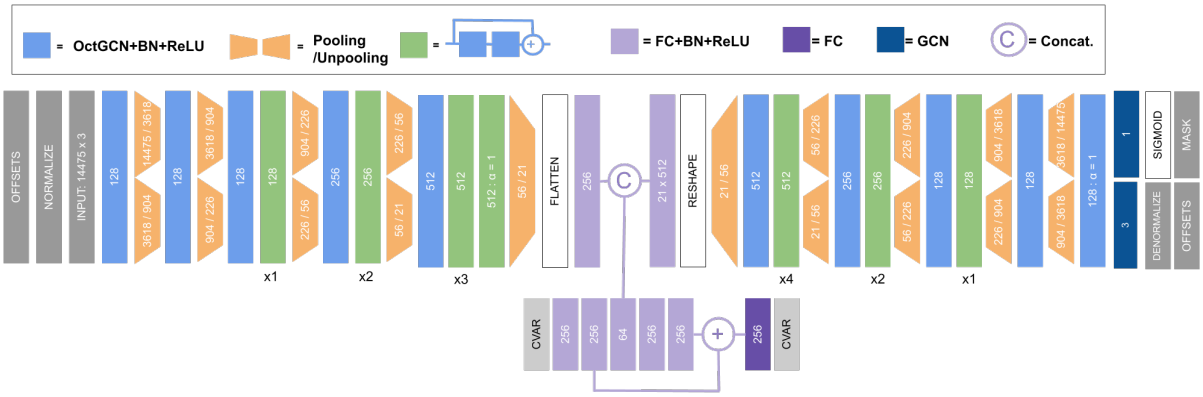

Figure 8: Modified Architecture to handle the high and low frequency components of octave convolution (OctGCN). When $\alpha = 1$ for the OctGCN layers, all of the features of the convolution are added to the high frequency component, and only the high frequency is returned. Otherwise, $\alpha = 0.125$, and both high and low frequency components are returned. The pooling is applied to both the high and low frequency components separately. The batch normalization (BN) and ReLU functions are applied to the high and low frequency components after each OctGCN layer.

of the encoder and decoder each output only the high frequency component. The low frequency component is tracked intermediately within these networks, but is not passed through the bottleneck. Figure 7 shows the architecture for the octave network.

4.4 Loss

During training of the networks, the loss is computed by two values: offsets, and normals. The offsets determine the predicted location of the garment vertices, and are provided as output from the network. The offset loss is then computed as the L1 loss between the ground truth and predicted offset values. The normals are computed by finding the normal vector perpendicular to each plane of the garment. The L1 loss between the predicted normals and ground truth normals is computed to determine the normal loss. The loss for the network is then computed as the summation of the offset loss and the normal loss.

5 Experiments

In this work we study different network architectures by performing modifications to the network described in section 3, which I shall refer to as the default network henceforth. In this section, I describe the experiments that I performed by applying the proposed changes in section 4 to the default network, and observing the results. I also explain the data and the network parameters used for training in sections 5.1 and 5.3, respectively. Finally, I present the results of an ablation study on the capacity of the network, as well as a comparison between the different networks tested.

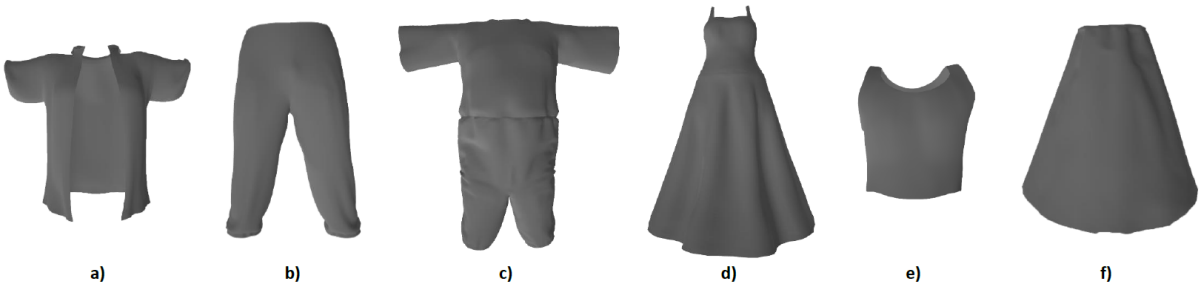

Figure 9: The six different garment types are: a) Tshirt, b) Trousers, c) Jumpsuit, d) Dress, e) Top, and f) Skirt.

5.1 Data

A subset of the CLOTH3D dataset was used for training and testing in this work. There were 647 outfits used for training, and 60 outfits used for validation. Each outfit consists of either one, or two garments (top and bottom, or one-piece), and each garment consists of approximately 300 different body poses, resulting in 194,835 training samples, and 24,395 validation samples. There are six different possible garment types for each outfit, shown in figure 9. Each of the garments also contains a fabric type from among: silk, denim, cotton, or leather. The fabric type is an important characteristic for physics-based simulations, as different fabrics fold/drape in different manners due to the toughness of the fabric (ie. leather would have fewer wrinkles than silk because leather is less pliable than silk).

5.2 Networks

There are five networks in this work that are trained and tested against one another. The first, is the default network in order to establish a baseline. Second, the CDC network which consists of replacing the vanilla convolution with the central difference convolution described in section 4.2. Thirdly, the pooling network, which uses the introduced spectral pooling from section 4.1. The fourth network is the octave network, which like the CDC network, simply replaces the vanilla convolution from the default network. Finally, the last network is an amalgamation of the CDC and the pooling networks, in order to observe the result of the combination of these techniques.

5.3 Training

Each network was trained for 52 epochs, with a learning rate of 0.0001, using adam optimizer, in batches of size 16. All experiments were completed on a single NVIDIA GeForce GTX 1080Ti graphics processor. The training was performed incrementally, starting with 200 training steps, and increasing by 200 every epoch. At each epoch, if the current offset loss for the validation samples is lower than the previous best, then the current model is saved, and the best updated. The training curves showing the offset loss for the validation set for each network are shown in figure 10. As it can be seen, both the default and pooling networks are not stable during training. However, each network was able to converge to nearly the same state. Furthermore, we can appreciate that the central difference convolution provided the most stable networks during training: CDC

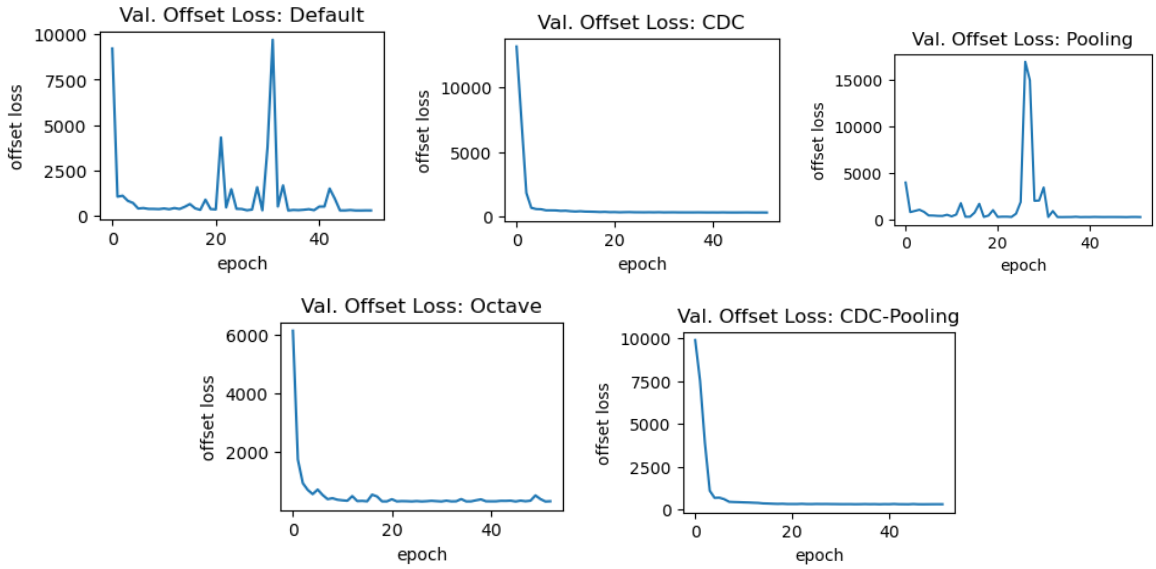

Figure 10: Loss training curves from the validation set during training, for each of the five main networks.

network and CDC-pooling.

5.4 Ablation Study

A small ablation study is performed on the capacity of the CDC network. As the CDC and default architectures are very similar (as described in section 4.2), we expect any changes in capacity to be reflected between these two networks. Thus, we chose the CDC network for this ablation study because the training curves in figure 10 show that the CDC network is more stable. In order to modify the capacity of the network, the number of features is adjusted. The default number of features in the final GCN layers before the bottleneck is 512, and decreases exponentially by a factor of two as the layers move away from the bottleneck. In this study, the number of features is increased to 1024, as well as decreased to 256, in order to observe the effect that either increasing or decreasing the capacity has on the network.

The training curves for the networks used in the ablation study are shown in figure 11. The 256 feature network had an initial lower loss than the 512 feature or 1024 feature networks, however, the loss decreased at a slower rate, with a minimum value of 292. Additionally, the increased capacity network with 1024 features also began with a lower loss value than the network with 512 features, and decreased at a slower rate, with a minimum loss of 286. Finally, the network with 512 features had the highest initial loss, decreased the fastest, and was able to achieve a minimum loss value of 284. Since the 512 feature network had the best performance, neither increasing nor decreasing the capacity of the network had a positive impact on the result.

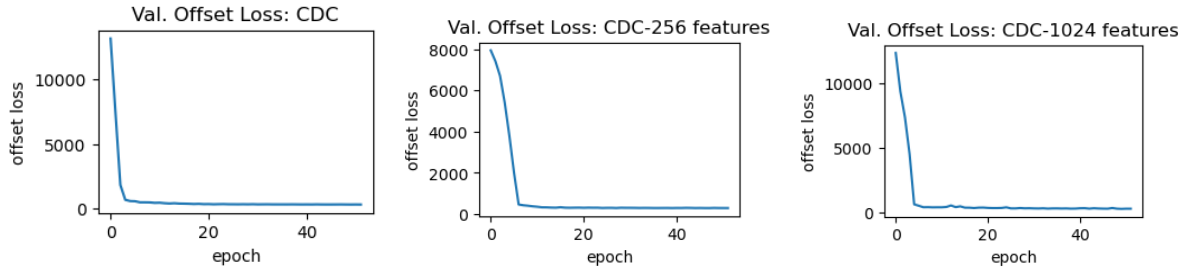

Figure 11: Loss training curves for the CDC network with alternative capacities. From left to right, the number of features in the final layer of each network are: 512, 256, and 1024.

5.5 Results

In this section, I discuss the results of the networks both quantitatively and qualitatively.

The quantitative results are computed using average per vertex Euclidean error of the garment vertices. These results are categorized into different blocks: per garment type, per fabric type, and overall quality. Table 1 shows the average error for each of these blocks, for each network. For each row in the table, corresponding to the different garment types or fabric types, the network that performed the best was either the CDC network, the pooling network, or the CDC-pooling network. This shows that both the CDC and the new pooling were both able to increase the performance of the network from the default. Furthermore, the total average of each network shows that the CDC and pooling networks both achieved the same, and best result over all of the networks. As stated, the CDC-pooling combined network performed best for certain fabric or garment types, but on average over all of the garments, it had a lower performance than the default network.

The octave network had the lowest performance out of all of the networks for almost all categories: garment or fabric types. Thus, the octave convolution hinders the performance of the garment reconstruction network.

As described in section 5.1, each garment, $G \in R^{F \times V \times 3}$ consists of different frames, F, corresponding to different body poses, and each frame contains a set of vertices, V corresponding to the garment in a particular pose. Figure 12 plots the ratio of the number of frames among all garments that have an average error less than a given distance threshold D. The variance of this value between the different networks is shown to be low, which means that all of the networks have approximately the same distribution of error among the samples. When the scale of the plot is increased, it is clear that the CDC network provides the lowest error at certain intervals, and the octave network has the most error. Upon further exploration of the graph by viewing alternative intervals, we found that the CDC, default, pooling, and CDC-pooling networks appear to frequently intertwine together, meaning that there is very little difference in the error ratios amongst these networks. The octave network does not at any point contain the lowest error.

The qualitative results are done by observing and comparing the rendered predicted output garments in different poses amongst the five networks. Figure 13 shows the cate-

	Default	CDC	Pooling	Octave	CDC-Pool
Tshirt	0.0282	0.0281	**0.0271**	0.0293	0.0282
Trousers	0.0245	0.0254	0.0254	0.0266	**0.0241**
Jumpsuit	0.0202	0.0204	**0.0201**	0.0210	0.0202
Dress	0.0375	**0.0371**	0.0374	0.0404	0.0381
Top	0.0212	0.0211	**0.0209**	0.0219	0.0230
Skirt	0.0639	**0.0610**	0.0624	0.0650	0.0641
silk	0.0397	0.0395	**0.0394**	0.0415	0.0408
denim	0.0306	**0.0303**	0.0304	0.0316	**0.0303**
cotton	0.0246	0.0248	**0.0245**	0.0262	0.0256
leather	0.0280	0.0270	0.0277	0.0297	**0.0266**
TOTAL	0.0304	**0.0302**	**0.0302**	0.0320	0.0308

Table 1: Average mean square error of the garment vertices for each of the networks. The average per garment type is computed first, followed by the average per fabric type, and lastly the total average for the network. The best result for each row is displayed in bold.

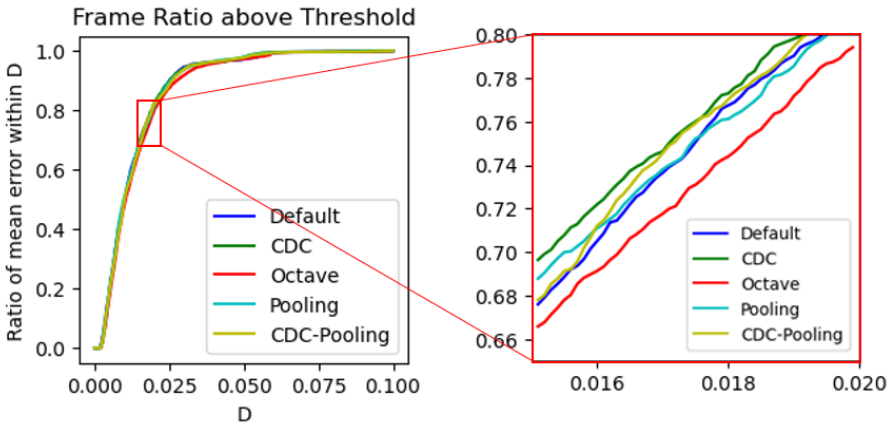

Figure 12: Ratio of the number of frames among all garments that have an error less than distance threshold $D \in [0.0001, 0.1]$. The right side shows a section of the graph at a larger scale.

gorized renders of select samples for each network, as well as the ground truth garments. In many cases of the predicted outputs, there are penetrations of the body through the garments, which occurs frequently due to the loss function not taking these intersections into account. All of the networks were fairly accurate at predicting the trousers, top, and tshirt garments. Even though some of the networks contained body-garment penetrations, the overall predicted garments were very similar to one another. On the other hand, the skirt and dress garments contain a large variety of predictions from among the networks.

Demonstrated in the fourth and seventh samples, it is clear that the pooling network has a higher capability of predicting the cloth dynamics for the skirt and dress garments than the other networks. While there was very little difference between the predicted garments for the tops, tshirts, trousers, and jumpsuits, the CDC-pooling network was able to obtain the fewest amount of visible penetrations, thereby making it the most suitable for predicting these garment types. Also, even though the pooling network claims the title for best prediction of the skirts and dresses, the CDC-pooling network is a close second, likely due to the pooling component in the network. Therefore, the CDC-pooling network has the overall highest qualitative performance among the networks.

6 Conclusions

In this work, we performed a study on different network architectures for 3D garment reconstruction. Five different architectures were trained using the CLOTH3D dataset, and compared against one-another. The default architecture that was used as a baseline for the study was a conditional variational auto-encoder, introduced in the CLOTH3D paper. The four remaining architectures in the study were all created by modifying certain aspects of the default. The modifications introduced were: 1) applying the central difference convolution (CDC) to GNNs 2) creating a new pooling method based on spectral clustering of the vertices in the SMPL human body mesh 3) applying the octave convolution to GNNs and 4) combining both the new pooling and the CDC. The results of the architectures were compared both quantitatively using the average error of the garment vertices, and qualitatively using rendered samples from each network.

The quantitative results have shown that the CDC and the new pooling networks were both able to outperform the default network, on average, and obtained an equivalent average error. The CDC-pooling network was able to achieve the lowest error for the trousers garment type, as well as two of the four fabric types among the garments. The octave network on the other hand performed the worst out of all five tested networks.

The qualitative results showed that all of the networks were able to predict the trousers, tshirt, and top garment types quite accurately, with very little difference between them. However, due to the nature of the skirt and dress topology, there exists more dynamics in the fabric - compared to the other topologies - making the skirt and dress more difficult to predict. The pooling network was clearly able to outperform the other networks for the dress and skirt garment types, as it was able to capture more of the proper cloth dynamics for these types.

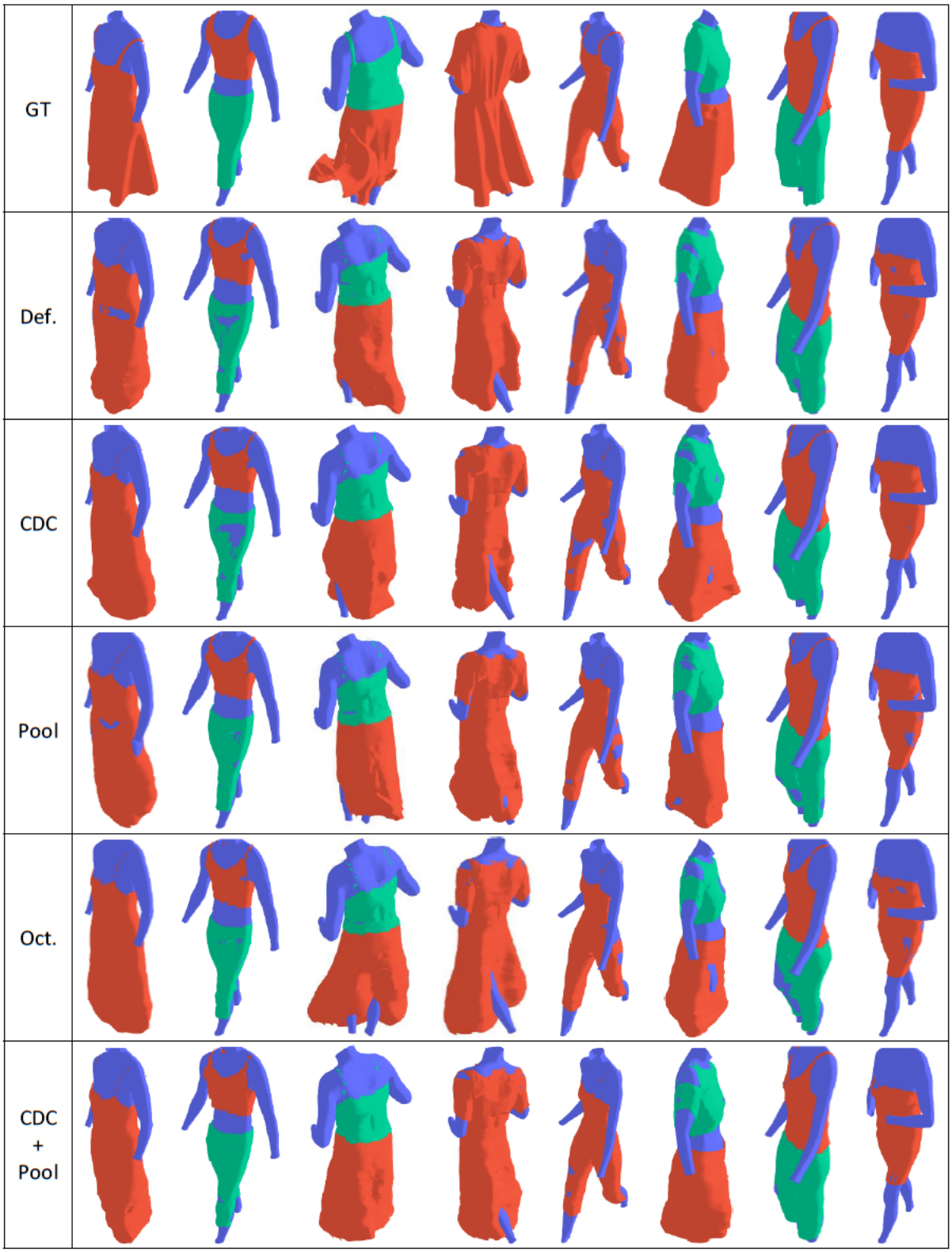

Figure 13: Rendered results from each network. First row represents the ground truth (GT) for the garments. The results of each network are demonstrated with the same set of samples, where each column represents a unique sample.

In conclusion, the study performed in this work showed that both the central difference convolution, and the new pooling based on spectral clustering provide slight improvement to the default network. Two considerations for future work are: 1) further exploration and improvement of the pooling, as this alteration had the largest impact on the correct modeling of the skirt and dress topologies and 2) modification of the loss function to penalize body penetrations in the garment.

References

[1] B. Amberg, S. Romdhani, and T. Vetter. "Optimal Step Nonrigid ICP Algorithms for Surface Registration". In: *2007 IEEE Conference on Computer Vision and Pattern Recognition*. 2007, pp. 1–8. DOI: 10.1109/CVPR.2007.383165.

[2] Hugo Bertiche, Meysam Madadi, and Sergio Escalera. "CLOTH3D: Clothed 3D Humans". In: *CoRR* abs/1912.02792 (2019). arXiv: 1912.02792. URL: http://arxiv.org/abs/1912.02792.

[3] Yunpeng Chen et al. "Drop an Octave: Reducing Spatial Redundancy in Convolutional Neural Networks with Octave Convolution". In: *CoRR* abs/1904.05049 (2019). arXiv: 1904.05049. URL: http://arxiv.org/abs/1904.05049.

[4] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. *Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering*. 2017. arXiv: 1606.09375 [cs.LG].

[5] Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge. *Texture Synthesis Using Convolutional Neural Networks*. 2015. arXiv: 1505.07376 [cs.CV].

[6] Pengpeng Hu et al. "3D High-quality Garment Reconstruction with Synthesized Texture". In: *Procedia Computer Science* 108 (2017). International Conference on Computational Science, ICCS 2017, 12-14 June 2017, Zurich, Switzerland, pp. 355–364. ISSN: 1877-0509. DOI: https://doi.org/10.1016/j.procs.2017.05.004. URL: http://www.sciencedirect.com/science/article/pii/S1877050917304945.

[7] Wenzel Jakob et al. "Instant Field-Aligned Meshes". In: *ACM Trans. Graph.* 34.6 (Oct. 2015). ISSN: 0730-0301. DOI: 10.1145/2816795.2818078. URL: https://doi.org/10.1145/2816795.2818078.

[8] Zorah Laehner, Daniel Cremers, and Tony Tung. *DeepWrinkles: Accurate and Realistic Clothing Modeling*. 2018. arXiv: 1808.03417 [cs.CV].

[9] Tony Lindeberg. "Scale-Space Theory in Computer Vision". In: 256 (1994). DOI: 10.1007/978-1-4757-6465-9.

[10] Matthew Loper et al. "SMPL: A Skinned Multi-Person Linear Model". In: *ACM Trans. Graph.* 34.6 (Oct. 2015). ISSN: 0730-0301. DOI: 10.1145/2816795.2818013. URL: https://doi.org/10.1145/2816795.2818013.

[11] Qianli Ma et al. "Learning to Dress 3D People in Generative Clothing". In: *arXiv e-prints*, arXiv:1907.13615 (July 2019), arXiv:1907.13615. arXiv: 1907.13615 [cs.CV].

[12] R. A. Newcombe et al. "KinectFusion: Real-time dense surface mapping and tracking". In: *2011 10th IEEE International Symposium on Mixed and Augmented Reality*. 2011, pp. 127–136. DOI: 10.1109/ISMAR.2011.6092378.

[13] Chaitanya Patel, Zhouyingcheng Liao, and Gerard Pons-Moll. *TailorNet: Predicting Clothing in 3D as a Function of Human Pose, Shape and Garment Style*. 2020. arXiv: 2003.04583 [cs.CV].

[14] Jianbo Shi and Jitendra Malik. "Normalized Cuts and Image Segmentation". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 22 (May 2002). DOI: 10.1109/34.868688.

[15] Zitong Yu et al. *Searching Central Difference Convolutional Networks for Face Anti-Spoofing*. 2020. arXiv: 2003.04092 [cs.CV].

[16] Bin Zhou et al. “Garment Modeling from a Single Image”. In: *Computer Graphics Forum* 32 (Oct. 2013). DOI: 10.1111/cgf.12215.

[17] Heming Zhu et al. *Deep Fashion3D: A Dataset and Benchmark for 3D Garment Reconstruction from Single Images*. 2020. arXiv: 2003.12753 [cs.CV].