UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Facultat d'Informàtica de Barcelona

Universitat de Barcelona
Facultat de Matemàtiques i Informàtica

ESCOLA TÈCNICA SUPERIOR
D'ENGINYERIA
Universitat Rovira i Virgili

Final Master's Degree Thesis

RGB to 3D garment reconstruction using UV map representations

Master's Degree in Artificial Intelligence

Author:
Albert Rial Farràs

Supervisors:
Sergio Escalera Guerrero
Meysam Madadi

June 2021

A Final Master's Degree Thesis submitted to the Facultat d'Informàtica de Barcelona (FIB) - Universitat Politècnica de Catalunya (UPC) - Barcelona Tech, Facultat de Matemàtiques de Barcelona - Universitat de Barcelona (UB) and Escola Tècnica Superior d'Enginyeria - Universitat Rovira Virgili (URV) in partial fulfillment of the requirements for the Master's Degree in Artificial Intelligence.

Thesis produced under the supervision of Prof. Sergio Escalera Guerrero and Prof. Meysam Madadi.

Author:
Albert Rial Farràs

Supervisors:
Sergio Escalera Guerrero
Meysam Madadi

Location:
Barcelona, Spain

Abstract

Predicting the geometry of a 3D object from just a single image or viewpoint is an intrinsic human feature extremely challenging for machines. For years, in an attempt to solve this problem, different computer vision approaches and techniques have been investigated. One of the domains in which there has been more research has been the 3D reconstruction and modelling of human bodies. However, the greatest advances in this field have concentrated on the recovery of unclothed human bodies, ignoring garments.

Garments are highly detailed, dynamic objects made up of particles that interact with each other and with other objects, making the task of reconstruction even more difficult. Therefore, having a lightweight 3D representation capable of modelling fine details is of great importance.

This thesis presents a deep learning framework based on Generative Adversarial Networks (GANs) to reconstruct 3D garment models from a single RGB image. It has the peculiarity of using UV maps to represent 3D data, a lightweight representation capable of dealing with high-resolution details and wrinkles.

With this model and kind of 3D representation, we achieve state-of-the-art results on CLOTH3D [4] dataset, generating good quality and realistic reconstructions regardless of the garment topology, human pose, occlusions and lightning, and thus demonstrating the suitability of UV maps for 3D domains and tasks.

KeyWords: Garment Reconstruction · 3D Reconstruction · UV maps · GAN · LGGAN · CLOTH3D · Computer Vision · Deep Learning · Artificial Intelligence.

Acknowledgments

First of all, I would like to express my gratitude to my supervisors Meysam and Sergio for the opportunity to work with their group. I am very grateful for the supervision and expertise of Meysam Madadi, established researcher of the Human Pose Recovery and Behavior Analysis (HuPBA) group. His implication, as well as all the help and advice provided, have been key for the development of the project. The results obtained are the fruit of countless meetings and discussions with him. This master thesis would also not have been possible without the direction of Sergio Escalera, expert in Computer Vision, especially in human pose recovery and human behaviour analysis.

I would also like to thank the entire Human Pose Recovery and Behavior Analysis (HuPBA) group and the Computer Vision Center (CVC) for providing me with the computational resources necessary for developing the project.

In the Master in Artificial Intelligence (MAI), I have met wonderful people, good friends and excellent professionals, who I am sure will achieve everything they aspire to do. I want to thank them for all the support they have given me during the development of this project and the whole master.

Finally, I cannot be more grateful to my family for their unconditional support in everything I do. They are my rock.

Contents

List of Figures

List of Tables

Glossary

CV:	Computer Vision
DL:	Deep Learning
AI:	Artificial Intelligence
GAN:	Generative Adversarial Network
CGAN:	Conditional Generative Adversarial Network
LGGAN:	Local Class-Specific and Global Image-Level Generative Adversarial Network
SMPL:	Skinned Multi-Person Linear model

1. Introduction

This master thesis tries to find a proper system to learn garment dynamics and reconstruct 3D garment models from single RGB images. We present a model based on Generative Adversarial Networks (GANs) [16] that has the peculiarity to use UV maps to represent 3D data. We study the feasibility of this type of representations for 3D reconstruction and generation tasks.

The GAN model used is based on LGGAN (Local Class-Specific and Global Image-Level Generative Adversarial Networks) [56], used previously on semantic-guided scene generation tasks like cross-view image translation and semantic image synthesis and characterised by combining two levels of generation, global image-level and local class-specific.

1.1 Motivation

Inferring 3D shapes from a single viewpoint is an essential human vision feature extremely difficult for computer vision machines. For this reason, there has always been a great deal of research devoted to building 3D reconstruction models capable of inferring the 3D geometry and structure of objects or scenes from a single or multiple 2D pictures. Tasks in this field go from reconstructing objects like cars or chairs to modelling entire cities. These models can be used for a wide variety of applications such as 3D printing, simulation of buildings in the civil engineering field, or generation of artificial objects, humans and scenes for videogames and movies.

One field in which the research community and industry have been working for years has been the study of human dynamics. Accurately tracking, capturing, reconstructing and animating the human body, face and garments in 3D are critical tasks for human-computer interaction, gaming, special effects and virtual reality. Despite the advances in the field, most research has concentrated only on reconstructing unclothed bodies and faces, but modelling and recovering garments have remained notoriously tricky.

For this reason, our work plans to push the research on the specific domain of garments, learning clothing dynamics and reconstructing clothed humans. As said, this is still a quite new research area, but with also many potential applications and benefits: allow virtual try-on experiences when buying clothes, reduce designers and animators workload when creating avatars for games and movies, etc.

First-generation methods that tried to recover the lost dimension from just 2D images were concentrated on understanding and formalising, mathematically, the 3D

to 2D projection process [20, 31]. However, this kind of solutions required multiple images captured with well-calibrated cameras and, in some cases, accurately segmented, which in many cases is not possible or practical.

Surprisingly, humans can easily estimate the approximate size and geometry of objects and imagine how they would be from a different perspective with just one image. We are able to do so because of all previously witnessed items and scenes, which have allowed us to acquire prior knowledge and construct mental models of how objects look like. The most recent 3D reconstruction systems developed have attempted to leverage this prior knowledge by formulating the problem as a recognition problem. Recent advances in Deep Learning (DL) algorithms, as well as the increasing availability of large training datasets, have resulted in a new generation of models that can recover 3D models from one or multiple RGB images without the complex camera calibration process. In this project, we are interested in these advantages, so we focus on developing a solution of this second category, using and testing different Deep Learning techniques.

In contrast to other models, our solution has the peculiarity of using UV maps as 3D representation. Compared to other representations such as meshes, point clouds or voxel-based representations, which are the ones commonly used in other 3D Deep Learning models [11, 13, 17, 19, 33, 38, 44, 45, 47, 57, 61–63, 65], UV maps allow us to use standard computer vision (CV) architectures, usually intended for images and two-dimensional inputs/outputs. In addition, and even more important, UV maps are lightweight and capable of modelling fine details, a feature necessary for modelling challenging dynamic systems like garments.

1.2 Contribution

The main contribution of this thesis is the proposal of a system that can adequately learn garment dynamics and reconstruct clothed humans from a single RGB image, using UV maps to represent 3D data and achieving state-of-the-art results in CLOTH3D [4] dataset.

In addition, this work also studies the feasibility and impact of using UV map representations for 3D reconstruction tasks, demonstrating that it is an interesting option that could be applied in many other tasks.

Finally, this project proposes and studies different strategies and techniques to perform this kind of task, and analyses quantitatively and qualitatively their contribution to our final solution.

1.3 Overview

Apart from this first chapter, where we introduced our work, the rest of the thesis is split into five more parts.

In the next chapter, Chapter 2, we detail all the required theoretical background needed to have a complete understanding of the project. In Chapter 3 we describe the related work and present similar works done in the current state-of-the-art.

Following, in Chapter 4 we extensively detail and present our proposal. Chapter 5 describes the dataset used for training and testing our model and shows all the experiments done and the results obtained, which are discussed and analysed quantitatively and qualitatively. Finally, in Chapter 6 we present the conclusions about this work and ideas for future work.

2. Background

This chapter presents the relevant background required to understand the followed methodology. First, it introduces the GAN architecture in which our model is based and then the different data representations usually used to represent 3D data, with an emphasis on UV maps, the representation used in our project.

2.1 Generative Adversarial Network

A Generative Adversarial Network (GAN) [16], designed in 2014 by Ian J. Goodfellow *et al.*, is a deep neural network framework that, given a set of training data, learns to generate new data with the same properties/statistics as the training data.

Indeed GAN architecture (Figure 2.1) is composed of two deep networks, a generator and a discriminator, which compete against each other.

- Generator: learns to generate realistic fake samples from a random seed. The generated instances produced are used as negative training examples for the discriminator.
- Discriminator: learns to distinguish real samples from fake ones (coming from the generator).

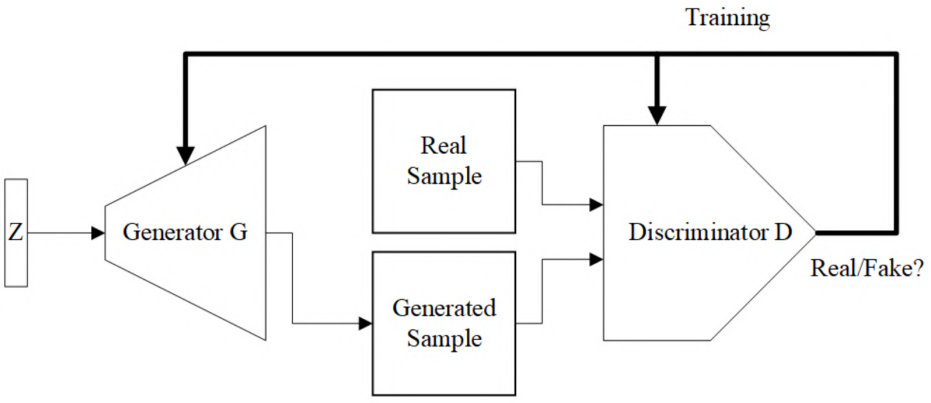


Figure 2.1: Generative Adversarial Network architecture. Source: [70]

The competition between both parts in this two-player game drives the whole model to improve until fake samples are good enough they cannot be distinguished

from real ones. To have a good generative model, both networks must perform well. If the generator is not good enough, it will not be able to fool the discriminator, and the model will never converge. On the other hand, if the discriminator performance is poor, then any sample will be labelled as real, even if it does not approximate the training data distribution, which means the model will never learn.

In the beginning, the generator's fake output is very easy for the discriminator to differentiate from the real data. Over time, the generator's output will become more realistic, and the generator will get better at fooling the discriminator. Eventually, the generator will generate such realistic outputs that the discriminator will be unable to distinguish them. In fact, for a perfect generator, the discriminator should have only 50% of accuracy, being completely random. Training the model beyond this convergence point could cause the discriminator to give wrong feedback to the generator, decreasing its performance.

To generate fake samples, the generator G receives as input some noise z, sampled using a normal or uniform distribution. Conceptually, this z represents the latent features of the samples generated, which are learned during the training process. Latent features are those variables that cannot be observed directly but are important for a domain. It can also be seen as a projection or compression of data distribution, providing high-level concepts of the observed data. This generator typically consists of multiple transposed convolutions that upsample vector z.

Once the synthetic samples $G(z)$ are generated, they are mixed together with real examples x coming from the training data and forwarded to the discriminator D, who classifies them as fake or real. The loss function of the discriminator is computed based on how accurate is its assessment and the hyperparameters are adjusted in order to maximise its accuracy. This loss is described formally in Equation 2.1, and it is composed of two terms: the first one rewards the recognition of real samples ($\log D(x)$), and the second one the recognition of generated samples ($\log(1 - D(G(z)))$). On the other hand, the generator is rewarded or penalised based on how well or not the generated samples fool the discriminator. Its objective function is shown in Equation 2.2, and it wants to optimise G so it can fool the discriminator D the most, generating samples with the highest possible value of $D(G(z))$. This two-player minimax game can also be formalised with just one single value function $V(G, D)$, shown in Equation 2.3, which the discriminator wants to maximise and the generator minimise.

$$\max_D V(D) = \mathbb{E}_{x \sim p_{data}(x)} \left[\log D\left(x\right)\right] + \mathbb{E}_{z \sim p_z(z)} \left[\log\left(1 - D\left(G\left(z\right)\right)\right)\right] \tag{2.1}$$

$$\min_G V(G) = \mathbb{E}_{z \sim p_z(z)} \left[\log\left(1 - D\left(G\left(z\right)\right)\right)\right] \tag{2.2}$$

$$\min_G \max_D V(D, G) = \mathbb{E}_{x \sim p_{data}(x)} \left[\log D\left(x\right)\right] + \mathbb{E}_{z \sim p_z(z)} \left[\log\left(1 - D\left(G\left(z\right)\right)\right)\right] \tag{2.3}$$

being $D(x)$ the estimated probability that an input example x is real, $\mathbb{E}_{x \sim p_{data}(x)}$ the expected value over all examples coming from training data, $G(z)$ the fake example produced by the generator for the random noise vector z, $D(G(z))$ the

estimate by the discriminator of the probability that a fake input example $G(z)$ is real, and $\mathbb{E}_{z\sim p_z(z)}$ the expected value over all random inputs to the generator.

These objective functions are learned jointly by the alternating gradient descent. In each training iteration, we first fix the generator weights and perform k training steps on the discriminator using generated samples, coming from the generator, and the real ones, coming from the training set. Then, we fix the discriminator parameters and we train the generator for a single iteration. We keep training both networks in alternating steps until we think the model has converged and the generator produces good quality samples. This training process can be formalised with the following pseudocode Algorithm 2.1:

Algorithm 2.1 GAN training

1: **for** number of training iterations **do**
2: **for** k steps **do**
3: Sample minibatch of m noise samples $z(1), ..., z^{(m)}$ from noise prior $p_g(z)$
4: Sample minibatch of m examples $x(1), ..., x^{(m)}$ from data generating distribution $p_{data}(x)$
5: Update the discriminator D by ascending its stochastic gradient:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D\left(x^{(i)}\right) + log\left(1 - D\left(G\left(z^{(i)}\right)\right)\right)\right]$$

6: **end for**
7: Sample minibatch of m noise samples $z(1), ..., z^{(m)}$ from noise prior $p_g(z)$
8: Update the generator G by descending its stochastic gradient:

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \left[\log\left(1 - D\left(G\left(z^{(i)}\right)\right)\right)\right]$$

9: **end for**

As during early training, the discriminator D typically wins against the generator G, the gradient of the generator tends to vanish and makes the gradient descent optimisation to be very slow. To speed it up, the GAN authors proposed to, rather than training G to minimise $\log(1 - D(G(z)))$, train it to maximise $\log D(G(z))$

2.1.1 Conditional GAN

Conditional Generative Adversarial Networks (CGANs) [39] are an extension of the GANs in which the generator and discriminator are conditioned by some additional input y. This new input can be of any kind (class labels or any other property) and is usually feed into both the discriminator and generator as an additional input layer, concatenating input noise z and condition value y, as can be seen in Figure 2.2.
With this modification, the objective function of the Conditional GAN can be expressed as follows:

$$\min_G \max_D V(D,G) = \mathbb{E}_{x\sim p_{data}(x)}\left[\log D\left(x|y\right)\right] + \mathbb{E}_{z\sim p_z(z)}\left[\log\left(1 - D\left(G\left(z|y\right)\right)\right)\right] \tag{2.4}$$

The motivation behind the conditioning is to have a mechanism to request the generator for a particular input. For example, suppose we train a GAN to generate

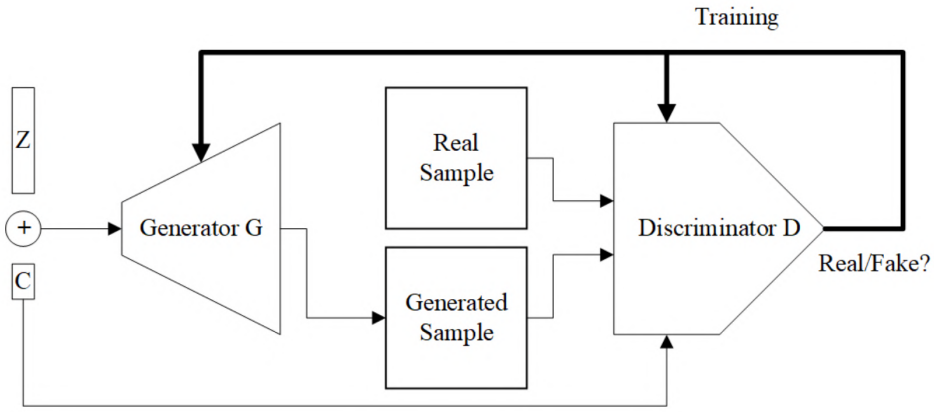


Figure 2.2: Conditional Generative Adversarial Network architecture. Source: [70]

new MNIST images [32], which is a very well-known dataset of handwritten digits. In that case, we will have no control over what specific digits will be produced by the generator. However, if we input to a Conditonal GAN the handwritten digit image together with its class label (number appearing in the image), the generator will learn to generate numbers based on their class and we will be able to control its output.

2.2 3D data representations

Computer Vision (CV) and Deep Learning (DL) research has typically focused on working with 2D data, images and videos. Recently, with the availability of large 3D datasets and more computational power, the study and application of Deep Learning techniques to solve tasks on 3D data has grown a lot. These tasks include segmentation, recognition, generation and reconstruction, among others. However, as there is no single way to represent 3D models, the 3D datasets available in the current state-of-the-art are found in a wide variety of forms, varying in both structure and properties.

In this section, we give an overview of the most common 3D representations used in Deep Learning architectures (point clouds, 3D meshes, voxels and RGB-D data) and the type of representation used and evaluated in this project (UV maps), detailing their differences and the challenges of each one.

2.2.1 Voxel-based representations

One voxel is basically a 3D base cubical unit that, together with other voxels, form a 3D object inside a regular grid in the 3D space. It can be seen as a 3D extension of the concept of pixel.

The main limitation of voxels is their inefficiency to represent high-resolution data, as they represent both occupied and non-occupied parts of the scene, needing an enormous amount of memory for non-occupied parts. Using a more efficient 3D volumetric representation such as octree-based, which consists basically of varying-sized voxels, one can represent higher resolution objects with less computation

consumption, thanks to their ability to share the same voxel for large regions of space. Despite this, both voxels and octree representations do not preserve the geometry of 3D objects, in terms of the shape and surface smoothness, and lack of enough detail, reason why they are not adequate in all domains.

2.2.2 Point clouds

A 3D point cloud is a set of unstructured data points in a three-dimensional coordinate system that approximates the geometry of 3D objects. These points are defined using x, y, z coordinates and are usually obtained by 3D scanners like LiDAR or photogrammetry [64].

Point-based representations are simple and efficient, but, as they are not regular structures, they do not fit into convolutional architectures that exploit spatial regularity. To get around this constraint, the following representations emerged:

- Point set representation, which treats a point cloud as a matrix of size $N \times 3$, being N the number of points.
- 3-channel grids of size $H \times W \times 3$, encoding in each pixel the x, y, z coordinates of a point.
- Depth maps from multiple viewpoints.

However, the lack of structure, caused by the absence of connectivity information between point clouds, results in an ambiguity about the surface information. In the domain of garments, this problem means that wrinkles cannot be handled properly, even if the number of points is very high.

2.2.3 3D meshes and graphs

Meshes are one of the most popular representations used for modelling 3D objects and shapes. A 3D mesh is a geometric data structure that allows the representation of surface subdivisions by a set of polygons. These polygons are called faces and are made up of vertices. The vertices describe how the mesh coordinates x, y, z exist in the 3D space and connect to each other forming the faces. Most commonly, these faces are triangles (triangular meshing), but there also exist quadrilateral meshes and volumetric meshes, which connect the vertices by tetrahedrons, pyramids, hexahedrons or prisms [64].

The challenge with this kind of representation is that Deep Learning algorithms have not been readily extended to such irregular representations. For this reason, it has become common to use graph-structured data to represent 3D meshes and use Graph Convolutional Networks (GCN) to process them. In these graphs, the nodes correspond to the vertices and the edges the connectivity between them. Using these graphs, has opened the door to the creation and innovation of Deep Learning models based on this kind of graph convolutions.

2.2.4 UV maps

A UV map is the flat representation of the surface of a 3D model, usually used to wrap textures easily and efficiently. The letters U and V refer to the horizontal

and vertical axes of the 2D space, as X, Y and Z are already used to denote the axis of the 3D space. Basically, in a UV map, each of the 3D coordinates/vertices of the object is mapped into a 2D flat surface.

Below, in Figure 2.3, we show an example of a 3D model (right) and its corresponding unwrapped surface on a UV map (left).

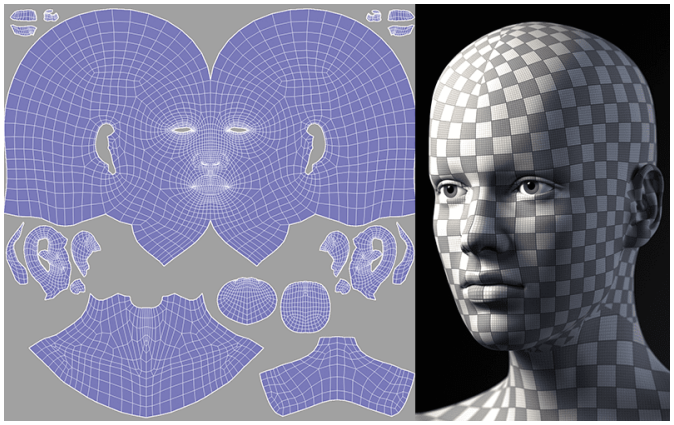


Figure 2.3: Example of a 3D face model and its corresponding UV map. Source: [58]

The process of creating a UV map is called UV unwrapping and consists of assigning to each 3D vertex (X, Y, Z) their corresponding UV coordinate. Thus, UV maps act as marker points that determine which points (pixels) on the texture correspond to which points (vertices) on the mesh. The inverse process, projecting the UV map onto a 3D model's surface, is called UV mapping.

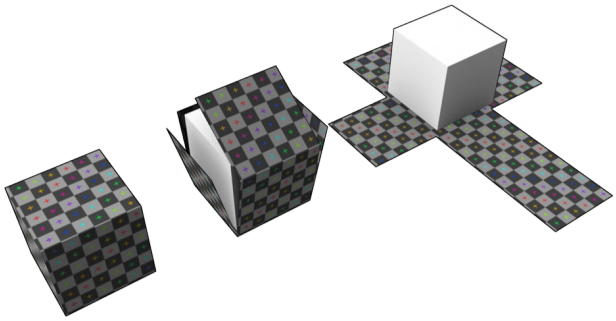


Figure 2.4: Representation of the UV mapping/unwrapping of a cube. Source: [58]

The main benefit of it is that it is a flat representation (2D), so it can be used in standard Computer Vision models thought for images or videos. Therefore one can take advantage of all the advances already done in Computer Vision for 2D data, which, as said before, is where there is more research and focus. Besides this, it is also a lightweight representation and can handle fine details, suitable for dealing with garment wrinkles.

3. Related work

After explaining and contextualising the necessary background, in this chapter we describe the works that are most related to our approach. First, we explain the leading works in the state-of-the-art of image-based 3D object reconstruction. Then we go into the specific domain of garments, presenting different solutions that reconstruct 3D garment models from single or multiple images. Finally, we give an overview of other works that also use and study UV map representations.

3.1 3D reconstruction

Fast and automatic 3D object reconstruction from 2D images is an active research area in Computer Vision with an increasing interest coming from many industries and applications: e-commerce, architecture, game and movie industries, automotive sector, 3D printing. For this reason, during the last decades, a large amount of work has been carried out on this topic, from reconstructing objects like chairs, cars or planes to modelling faces, scenes, or even entire cities.

The first works done approached the problem from a geometric perspective and tried to understand and formalise the 3D projection process mathematically. It is the case of techniques based on the multi-view geometry (MVG) [20], also known as stereo-based techniques, that match features across the images from different views and use the triangulation principle to recover 3D coordinates. Although these solutions are very successful in some scenarios such as structure from motion (SfM) [53] for large-scale high-quality reconstruction and simultaneous localisation and mapping (SLAM) [9] for navigation, they are subject to several restrictions: they need to have a great number of images from different viewpoints of the object, and this cannot be non-lambertian (*e.g.* reflective or transparent) nor textureless. If the number of images is not enough or they do not cover the entire object, MVG will not reconstruct the unseen parts as it will not be able to establish feature correspondences between images. The lack of textures and reflections also affect this feature matching [5, 51].

With the aim of being able to reconstruct non-lambertian surfaces, shape-from-silhouette, or shape-by-space-carving, methods [31] appeared and became popular. However, these methods require the objects to be accurately segmented from the background or well-calibrated cameras, which is not suitable in many applications.

All these restrictions and disadvantages lead the researchers to draw on learning-based approaches, that consider single or few images and rely on the shape prior knowledge learnt from previously seen objects. Early works date back to Hoiem

et al. [22] and Saxena *et al.* [52]. Nevertheless, it was not until recently, with the success of Deep Learning architectures, and more importantly, the release of large-scale 3D datasets such as ShapeNet [10], that learning-based approaches achieved great progress and very exciting and promising results.

This section provides an overview of some of the most important and well-known 3D reconstruction models belonging to this latter family.

3.1.1 Learning-based approaches

As explained in Chapter 2, unlike 2D images, which are always represented by regular grids of pixels, 3D shapes have various possible representations, being voxels, point clouds and 3D meshes the most common. This has led to 3D reconstruction models that are very different from each other, as their architectures largely depend on the representation used. For this reason, we review the most relevant works of each of these types of representation, giving an overview of their architecture and characteristics.

3.1.1.1 Voxel-based

Voxel-based representations were among the first representations used with Deep Learning techniques to reconstruct 3D objects from images. This is probably due to their 2D analogy, pixels, a data type that has been used in computer vision for decades. Because of this, during the last few years, many different approaches dealing with this representation have appeared, each one attempting to outperform the preceding and pushing the state-of-the-art.

3D-R2N2 3D Recurrent Reconstruction Neural Network, usually called just 3D-R2N2 [11] is a recurrent neural network architecture designed by Choy *et al.* that unifies both single and multi-view 3D object reconstruction problems. The framework takes in one or more images of an object from different perspectives to learn a 3D reconstruction of it. The network can perform incremental refinements and adapt as new viewpoints of the object become available. It is an encoder-decoder architecture composed of mainly three modules: a 2D Convolutional Neural Network (2D-CNN), a 3D Convolutional LSTM (3D-LSTM) and a 3D-Deconvolutional Neural Network (3D-DCNN). The 2D-CNN is in charge of encoding the input images into features. Then, given these encoded features, a set of 3D-LSTM units selects either to update their cell states or retain them by closing the input gate, allowing the network to update the memory with new images or retain what it has already seen. Finally, the 3D-DCNN decodes the hidden states of the 3D-LSTM units, generating a 3D probabilistic voxel reconstruction.

3D-VAE-GAN With the success of GANs [16] and variational autoencoders (VAEs) [27], Wu *et al.* [63] presented 3D-VAE-GAN, inspired by VAE-GAN [30]. This model applies GANs and VAEs in voxel data to generate 3D objects from just one single-view image. It is composed of an encoder that infers a latent vector z from the input 2D image and a 3D-GAN, which learns to generate the 3D object from the latent space vector z by using volumetric convolutional networks.

Octree Generating Networks Due to the high computational cost of voxel-based representations, the methods described before cannot cope with high-resolution models. For this reason, Tatarchenko *et al.* [57] proposed the Octree Generating Networks (OGN), a deep convolutional decoder architecture that can generate volumetric 3D outputs in an efficient manner by using octree-based representations, being able to manage relatively high-resolution outputs with a limited memory budget. In OGN, the representation is gradually convolved with learnt filters and up-sampled, as in a traditional up-convolutional decoder. However, the novelty of it is that, starting from a certain layer in the network, dense regular grids are replaced by octrees. As a result, the OGN network predicts large regions of the output space early in the first decoding stages, saving computation for the subsequent high-resolution layers.

Pix2Vox Recently, to overcome the limitations of solutions that use recurrent neural networks (RNNs) like 3D-R2N2 [11], Xie *et al.* [65] introduced Pix2Vox model. The main restrictions of models based on recurrent networks are the time consumption, since input images cannot be processed in parallel, and the inconsistency between predicted 3D shapes of two sets containing the same input images but processed in a different order. For this reason, the authors of Pix2Vox proposed a model composed of multiple encoder-decoder modules running in parallel, each one predicting a coarse volumetric grid from its input image frame. These different predicted coarse 3D modules are merged in a fused reconstruction of the whole object by a multi-scale context-aware fusion model, which selects high-quality reconstructions. Finally, a refiner corrects the wrongly recovered parts generating the final reconstruction.

3.1.1.2 Point-based

Similar to the works using volumetric-based representations, models that use point-based representations follow an encoder-decoder model. In fact, all of them use the same architecture for the encoder but differ in the decoder part. The works that use point sets to represent point clouds use fully connected layers on the decoder, since point clouds are unordered, while the ones that use grids or depth maps use up-convolutional networks.

Point Set Generation Network Fan *et al.* [13] presented a network that combines both point set and grid representations, composed of a cascade of encoder-decoder blocks. Each block takes the output of its previous block and encodes it into a latent representation that is then decoded into a 3-channel image. The first block takes as input the image. The last block has an encoder followed by a predictor of two branches: a decoder that predicts a 3-channel image, being each pixel the coordinates of a point, and a fully-connected network that predicts a matrix of size $N \times 3$, being each row a 3D point. To output the final prediction, the predictions of both branches are merged using a set union.

DensePCR As the network mentioned above and its variants are only able to recover low-resolution geometry, Mandikal *et al.* [38] presented DensePCR, a

framework composed of a cascade of multiple networks capable of handling high resolutions. The first network of the cascade predicts a low-resolution point cloud. Then, each subsequent block receives as input the previously predicted point cloud and computes global features, using a multi-layer perceptron architecture (MLP) similar to PointNet [44] and PointNet++ [45], and local features, using MLPs in balls surrounding each point. These global and local features are combined and fed to another MLP, which predicts a dense point cloud. This procedure can be repeated as many times as necessary until the desired resolution is achieved.

3.1.1.3 Mesh-based

The last most common used representation in 3D reconstruction models are 3D meshes. Compared to point-based representations, meshes contain connectivity information between neighbouring points, making them better for describing local regions on surfaces.

Pixel2Mesh Wang *et al.* proposed Pixel2Mesh [61], an end-to-end Deep Learning architecture designed to produce 3D triangular meshes of objects given just a single RGB image. Their approach is based on the gradual deformation of an ellipsoid mesh with a fixed size to the target geometry. To do so, they use Graph-based Convolutional Networks (GCNs) and refine the predicted shape gradually using a coarse-to-fine fashion, increasing the number of vertices as the mesh goes through the different stages. This network outperforms 3D-R2N2 framework, as the latter lacks of details due to the low resolution that voxel-based representations allow.

Wen *et al.* [62] extended Pixel2Mesh and proposed Pixel2Mesh++, which basically allows reconstructing the 3D shapes from multi-view images. To do so, Pixel2Mesh++ presents a Multi-view Deformation Network (MDN) that incorporates the cross-view information into the process of mesh generation.

AtlasNet Another generative model for mesh-based representations is AtlasNet, proposed by Groueix *et al.* [17], but with a totally different approach. AtlasNet decomposes the surface of a 3D object into m patches and learns to convert 2D square patches into 2-manifolds to cover the surface of the 3D shape (Papier-Mâché approach) by using a simple Multi-Layer Perceptron (MLP). The designed decoder is composed of m branches, and each branch is in charge of reconstructing a different patch. Then, these reconstructed patches are merged together to form the entire surface. Its main limitation is, however, that the number of patches m is a parameter of the system and its optimal number depends on each surface and object, making the model not general enough.

3.2 Garment reconstruction

If inferring 3D shapes of simple and static objects like chairs or cars from a single image is already complicated, doing so for 3D cloth models is extremely challenging. Garments are not only very dynamic objects but also belong to a domain where there is a huge variety of topologies and types.

Early works focused on garment modelling from images approached the problem from a geometric perspective [8,24,68]. Then, the advent of deep learning achieved impressive progress in the task of reconstructing unclothed human shape and pose from multiple or single images [7,25,28,29,42,43,54,66]. Nevertheless, recovering clothed human 3D models did not progress in the same way, not only because of the challenges mentioned before but also as a consequence of the lack of large annotated training datasets available.

It is only recently, in the last few years, that new large datasets and deep learning works focused on this domain started to emerge, trying to push its state-of-the-art. A standard practice used in garment reconstruction frameworks is to represent garments as an offset over the Skinned Multi-Person Linear model (SMPL) body mesh [1,6], which is a skinned vertex-based model that accurately represents a wide variety of body shapes in natural human poses [34]. However, this type of approach has the disadvantage that it may fail for loose-fitting garments, which have significant displacements over the shape of the body. For this reason, there are works that use non-parametric representations such as voxel-based representations [60], visual hulls [40] or implicit functions [12,49,50]. Finally, there are also some works that have combined both parametric and non-parametric representations [23,67,69].

Below, we analyse the characteristics and contributions of some of these deep learning frameworks.

Multi-Garment Net Multi-Garment Network (MGN) [6] designed by Bhatnagar *et al.* in 2019 is a network that reconstructs body shape and clothing, layered on top of the SMPL model, from a few frames of a video. It was the first model that was able to recover separate body shape and clothing from just images.

The model introduces class-specific garment networks, each one dealing with a particular garment topology. Despite this, because of the very limited dataset used, presented in the same work, with just a few hundreds of samples, each branch is prone to overfitting issues. Furthermore, as MGN relies on pre-trained parametric models, it is unable to handle out-of-scope deformations. Moreover, it typically requires eight frames as input to generate good quality reconstructions.

Octopus Also, in 2019, Alldieck *et al.* [1] presented Octopus, a learning-based approach to infer the personalised 3D shape of people, including hair and clothing, from few frames of a video.

It is a CNN-based method that encodes the frames of the person (in different poses) into latent codes that then are fused to obtain a single shape code. This code is finally fed into two different networks, one that predicts the SMPL shape parameters and other that predicts the 3D vertex offsets of the garment.

As the Multi-garment net framework [6], Octopus expresses the garment as an offset over the SMPL body mesh of the human, approach that, as mentioned before, fails in some types of garments that are not tight to the body, like dresses or skirts.

PIFu To be able to handle these clothes that largely depart from the body shape defined by the SMPL, Pixel-aligned Implicit Function (PIFu) [49] authors presented a kind of representation based on implicit functions that locally aligns pixels of 2D images with the global context of their corresponding 3D object. This representation does not explicitly discretise the output space but instead regresses a function that determines the occupancy for any given 3D location. This approach can handle high-resolution 3D geometry without having to keep a discretised representation in memory.

In this work, the authors also proposed an end-to-end deep learning method that uses PIFu representation for digitising highly detailed clothed humans that can infer both 3D surface and texture from one or multiple images.

More recently, the same authors presented PIFuHD [50], a multi-level framework also based on PIFu representation, which can infer 3D geometry of clothed humans at an unprecedentedly high 1k image resolution.

Deep Fashion3D In Deep Fashion3D work [69], the authors presented a novel large dataset to push the research in the task of garment reconstruction from just one single image. Apart from the dataset, the authors proposed a novel reconstruction framework that takes advantage of both meshes and implicit functions by using a hybrid representation.

As the methods based on implicit functions showed a great ability to manage fine geometric details, authors proposed transferring the high-fidelity local details learned from them to a template mesh, which adapts its topology to fit the input image's clothing category has robust global deformations.

By using this combined representation, the model is able to handle finer details and out-of-scope deformations.

3.3 UV map works

Within the research of garment modelling, there are a few works that have already used UV map representations in some way. For this reason, we briefly analyse their approach, how they use UV maps and for what purpose.

DeepCloth DeepCloth [55] is a neural garment representation framework which goal is to perform smooth and reasonable garment style transition between different garment shapes and topologies.

To do so, they use a UV-position map with mask representation, in which the mask denotes the topology and covering areas of the garments, while the UV map denotes the geometry details, *i.e.* the coordinates of each vertex. DeepCloth maps both the UV map and its mask information into a feature space using a CNN-based encoder-decoder structure. Then, by changing and interpolating these features, garment shape transition and editing can be performed.

Tex2Shape Alldieck *et al.* [2] presented Tex2Shape, a framework that aims to reconstruct high-quality 3D geometry from an image by regressing displacements in an unwrapped UV space.

In order to do so, the input image is transformed into a partial UV texture using DensePose [18], mapping input pixels to the SMPL model, and then translated by Tex2Shape network into normal and displacement UV maps. The normal map contains surface normals, while the vector displacement map contains 3D vectors that displace the underlying surface. Both normals and displacements are defined on top of the body SMPL model.

With this workflow, the reconstruction problem is turned into an easier 3D pose-independent image-to-image translation task.

SMPLicit SMPLicit [12], a concurrent work to ours, is a novel generative model designed by Corona *et al.* to jointly represent body pose, shape and clothing geometry for different garment topologies, while controlling other garment properties like garment size.

Its flexibility builds upon an implicit model conditioned with the SMPL body parameters and a latent space interpretable and aligned with the clothing attributes, learned through occlusion UV maps.

The properties that the model aims to control are: clothing cut, clothing style, fit (tightness/looseness) and the output representation. To learn the latent space of the first two properties, clothing cut and style, the authors compute a UV body occlusion map for each garment-body pair in the dataset, setting every pixel in the SMPL body UV map to 1 if the body vertex is occluded by the garment, and 0 otherwise.

4. Methodology

In this chapter, we present the methodology followed to build our system and reconstruct 3D garment models. First, we explain how we use UV map representations in our approach, and the pre-processing techniques and pipeline carried out to transform them into usable and suitable inputs for our model, a step that has proven to be of enormous importance. After that, we detail our specific model, explaining the architecture in which it is based and the modifications made to support our task.

4.1 Our UV maps

As explained in Chapter 2, UV maps are the flat representation of the surface of a 3D model. They are usually used to wrap textures, being able to project any pixel from an image to a point on a mesh surface. Nevertheless, as seen in previous Chapter 3, UV maps can also be used to denote other properties such as normal coordinates or vertices coordinates, since they can assign any property encoded in a pixel to any point on a surface.

In particular, in our approach, we use UV maps that encode garment mesh vertices, storing 3D coordinates as if they were pixels of an RGB image, together with a mask that denotes the topology and covering areas of the garment. In this way, we achieve a kind of representation that can be used in standard computer vision frameworks, usually thought for 2D data like images and videos, leveraging all progress done in this field. Moreover, unlike other representations such as volumetric representations, UV maps are perfectly capable of handling high-frequency cloth details efficiently.

This type of representation is also used in DeepCloth [55] and Tex2Shape [2] frameworks, which refer to it as *"UV-position map with mask"* and *"UV shape image"* respectively.

4.1.1 Normalise UV maps

As explained above, our UV maps contain 3D coordinate vectors x, y, z that are mapped to the garment mesh vertices. These coordinates do not have a fixed range and can be highly variable. For this reason, since large variability on the data can be a problem during the training of networks, getting stuck in local minima, a common practice is to normalise or standardise these values, helping the model to have a more stable and efficient training.

To normalise our UV maps, we propose two different approaches:

1. **Min-max normalization**: the first proposal is to use the minimum and maximum values of coordinates $x_{\min}, y_{\min}, z_{\min}$ and $x_{\max}, y_{\max}, z_{\max}$ within the training dataset to apply a min-max normalization and normalise all values between the range of 0 and 1.

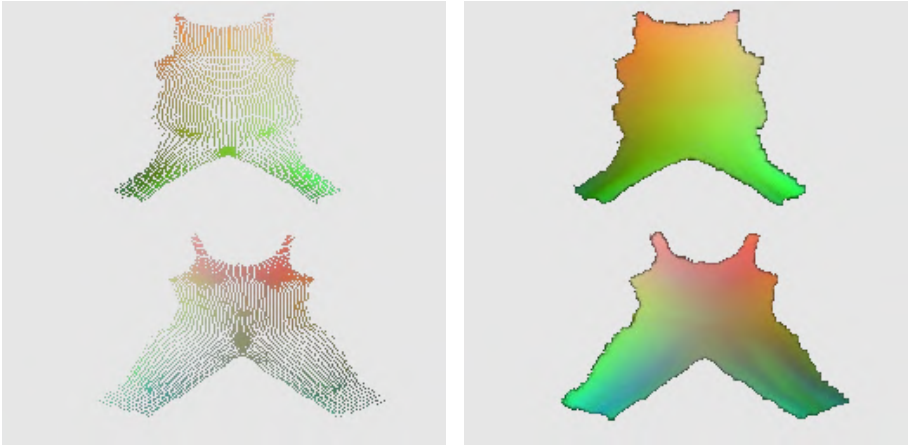
2. **Displacement UV maps**: the other proposal is to do the same as other approaches like [1, 2, 6] and represent garment vertices as an offset over the estimated SMPL body vertices. Alldieck *et al.* [2] refer to this kind of UV maps as displacement maps, because they displace the underlying body surface. With this approach, we somehow standardise the values and obtain a much lower variability.

4.1.2 Inpainting

A 3D mesh is a discrete representation, as it has a finite number of vertices. Therefore, the UV maps we use are also discrete, as they are the 2D projection of those vertices. For this reason, the UV maps have empty gaps between vertices, as can be seen in Figure 4.1a.

To avoid these gaps and make the UV map image smoother, we use image inpainting techniques to estimate the values of the empty spaces. Image inpainting is a form of image restoration that is usually used to restore old and degraded photos or repair images with missing areas. In particular, we use the Navier-Stokes based method [3], which propagates the smoothness of the image via partial differential equations and at the same time preserves the edges at the boundary [37]. With this technique, we obtain a UV map like the one in Figure 4.1b.

In Chapter 5 we detail the results of an experiment that analyses the contribution of this pre-processing step.



(a) Original UV map (b) Inpainted UV map

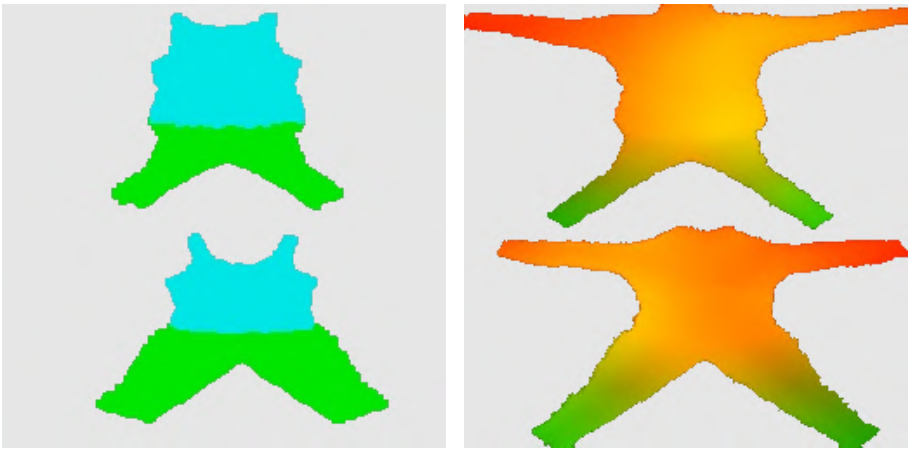
Figure 4.1: Comparison between the original UV map and the inpainted one, filling the empty gaps between projected vertices.

4.1.3 Semantic map and SMPL body UV map

Last but not least, as input, the LGGAN model expects not only an RGB image but also a condition that guides and controls it through the generation. For this purpose, for each UV map, we estimate a semantic segmentation map S that contains the topology of the garments and segment them in different labels/classes, one per garment type. An example is shown in Figure 4.2a.

Another option used to condition the model is the SMPL body UV map. This body UV map is obtained in the same way we obtain garments UV maps, by projecting the SMPL body mesh into the 2D UV space and applying the same pre-processing techniques. Figure 4.2b shows an example of it. The SMPL body surface is inferred by using SMPLR [36], a deep learning approach for body estimation.

With these conditions, the model can not only reconstruct 3D garments but also transition between different body and garment shapes and topologies.



(a) Semantic segmentation map S (b) SMPL body UV map B

Figure 4.2: Examples of the semantic segmentation maps and the SMPL body UV maps used to condition our model. In Figure 4.2a each color represents a different garment type.

4.2 LGGAN for 3D garment reconstruction

Investigating what solution we could propose to learn garment dynamics and reconstruct 3D garment models from single RGB images through UV map representations, we first thought that our task could be seen as an image-to-image translation problem. The models used for this kind of tasks learn mappings from input images to output images. This fits our case since our input is also an image and, although our expected output is a UV map, this has the same dimensionality as an RGB image: $H \times W \times 3$.

However, in these image-to-image translation systems, the input and output have pixel correspondence between them, while in our case pixels follow a very different path from the RGB image (input) to the UV map (output). For this reason, we searched for systems covering the cross-view image translation task proposed in [46], where this pixel correspondence between input and output does not exist

either. This task's main goal is to generate a ground-level/street-view image from an image of the same scene but from an overhead/aerial view, and vice versa. It is a very challenging task since the information from one view usually tells very little about the other. For example, an aerial view of a building, *i.e.* the roof, does not give much information about the colour and design of the building seen from the street-view.

As standard image-to-image translation models do not perform well on this task, recently, several specific models have emerged trying to solve this cross-view image synthesis task. One of the best works on this topic is the one published in 2020 by Tang *et al.* [56], in which the Local class-specific and Global image-level Generative Adversarial Networks (LGGAN) model was presented, system in which we draw from and adapt for our domain.

4.2.1 Architecture

The LGGAN proposed by Tang *et al.* [56] is based on GANs [16], more specifically in CGANs [39], and is mainly composed of three parts/branches: a semantic-guided class-specific generator modelling local context, an image-level generator modelling the global layout, and a weight-map generator for joining the local and the global generators. An overview of the whole framework is shown in Figure 4.3.

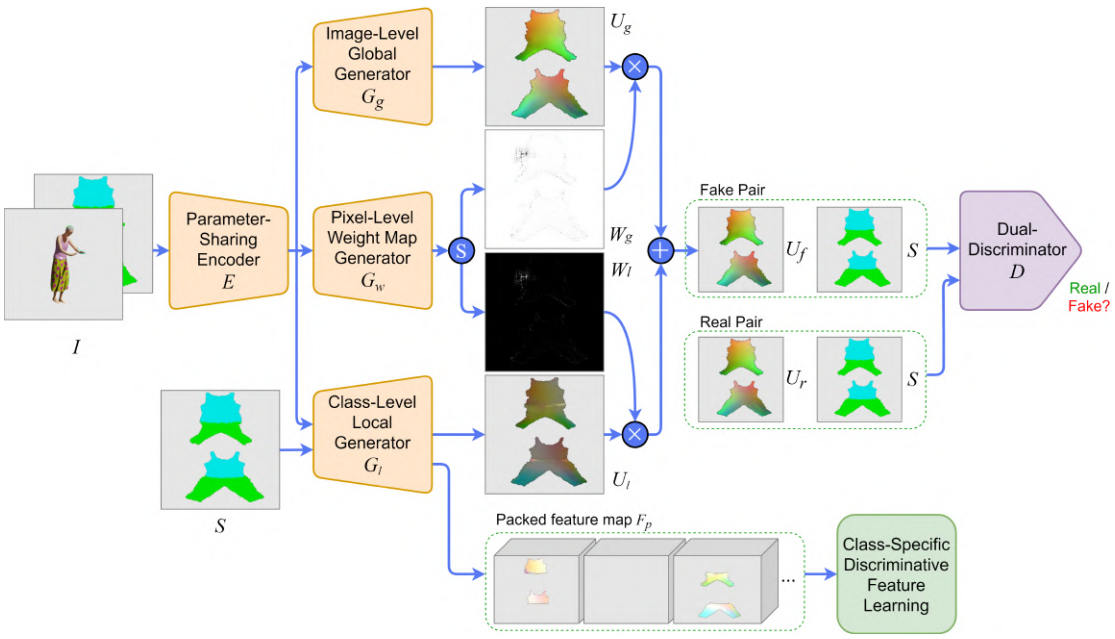


Figure 4.3: Overview of the proposed LGGAN. The symbol $\oplus$ denotes element-wise addition, $\otimes$ element-wise multiplication and $\textcircled{s}$ channel-wise softmax. Source: Own elaboration adapted from [56].

Like the original GAN [16], LGGAN is composed of a generator G and a discriminator D. The generator G consists of a parameter-sharing encoder E, an image-level global generator G_g, a class-level local generator G_l and a weight map generator G_w. The encoder E shares parameters to all the three generation branches to make a compact backbone network. Gradients from all the three generators G_g,

G_l and G_w contribute together to the learning of the encoder. In the original model, the input RGB image is concatenated with a semantic map and fed to this backbone encoder E. This semantic map is a semantic segmentation of the target image, with a class label for every type of object appearing on it. By adding this map, the authors somehow guide the model to learn the correspondences between the target and source views. As explained before, in our case, we present two different options to guide the reconstruction. The first one is conditioning in the same way they do in the original model, by concatenating to the input RGB image a semantic segmentation map S of the target UV map, containing a different label/class for each garment type. The other option proposed is to condition the class-level local generator G_l on this UV map segmentation S, but use the SMPL body UV map B of the person to condition the image-level global generator G_g and the discriminator D. Figure 4.2 shows examples of these inputs. Figure 4.4 presents the same LGGAN overview shown before but when conditioning also on the SMPL body UV map B.

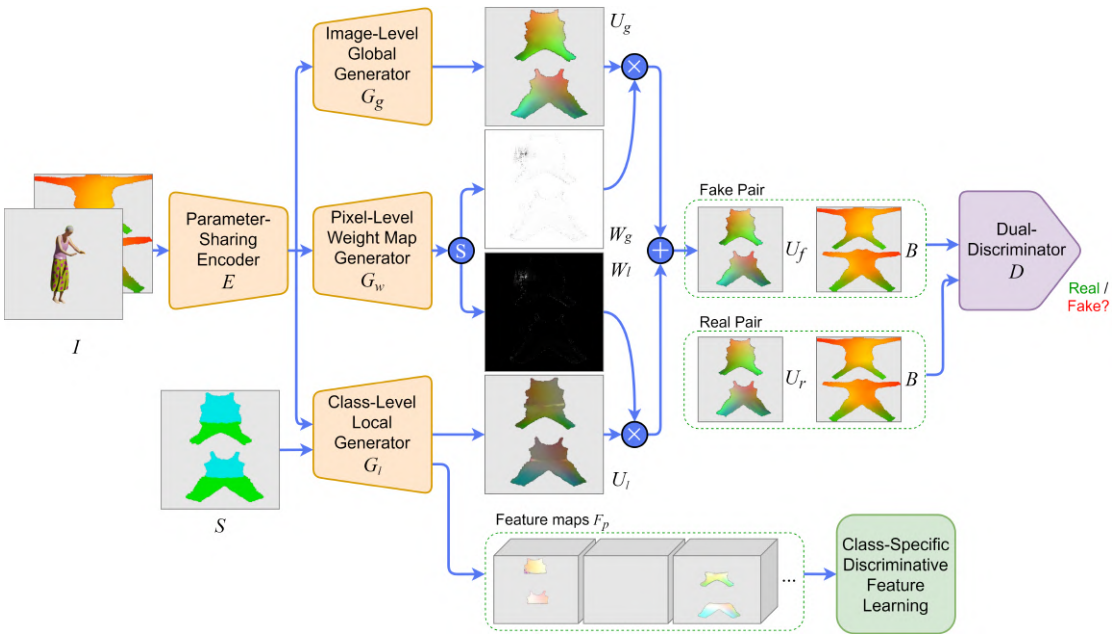


Figure 4.4: Overview of the proposed LGGAN when conditioning also on SMPL body UV map B. The symbol $\oplus$ denotes element-wise addition, $\otimes$ element-wise multiplication and Ⓢ channel-wise softmax. Source: Own elaboration adapted from [56].

LGGAN extends the standard GAN discriminator to a cross-domain structure that receives as input two pairs, each one containing a UV map and the condition of the model. Nevertheless, the goal of the discriminator D is the same, try to distinguish the generated UV maps from the real ones.

Finally, apart from the generator and discriminator, there is also a novel classification module responsible for learning a more discriminative and class-specific feature representation.

4.2.1.1 Generation branches

Parameter-Sharing Encoder The first module of the network is a backbone encoder E. This module basically takes the input I and applies several convolutions to encode it and obtain a latent representation $E(I)$. Its architecture is presented in 4.5. It is composed of three convolutional blocks and nine residual blocks (ResBlocks) [21]. Both blocks are similar and consist of convolutional layers, instance normalisation layers and ReLU activation functions. The only difference is that ResBlocks contain also skip connections.

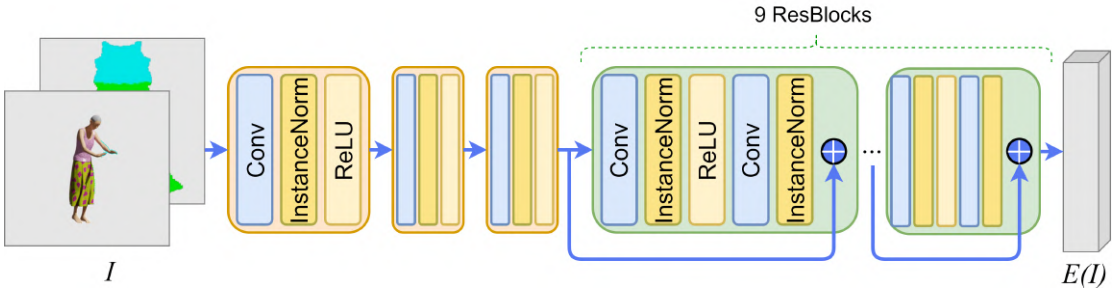


Figure 4.5: Parameter-Sharing Encoder architecture. Contains three convolution blocks and nine residual blocks. Source: Own elaboration.

Note that the input I, as explained before, is the concatenation of the input RGB image and the condition. This condition could be either the UV map semantic guide S or the SMPL body UV map B.

Class-Specific Local Generation Network The authors propose a novel local class-specific generation network G_l that separately constructs a generator for each semantic class. Each sub-generation branch has independent parameters and concentrates on a specific class, producing better generation quality for each class and yielding richer local details. Its overview is shown in Figure 4.6.

This generator receives as input the output of the encoder, the encoded features $E(I)$. These are fed into two consecutive deconvolutional blocks to increase the spatial size. These deconvolutional blocks consist of a transposed convolution followed by an instance normalisation layer and a ReLU activation, as shown in Figure 4.7a. The scaled feature map f' is then multiplied by the semantic mask of each class M_i, obtained from the semantic map S. By doing this multiplication, we obtain a filtered class-specific feature map for each one of the classes. This mask-guided feature filtering can be expressed as:

$$F_i = M_i \cdot f', \quad i = 1, 2, \ldots, c \tag{4.1}$$

where c is the number of semantic classes (*i.e.* number of types of garment). After computing these filtered feature maps, each feature map F_i is fed into a different convolutional block, the one for the corresponding class i, which generates a class-specific local UV map U_{l_i}. Each convolutional block is composed of a convolutional layer and a Tanh activation function, as shown in Figure 4.7b. The loss used in

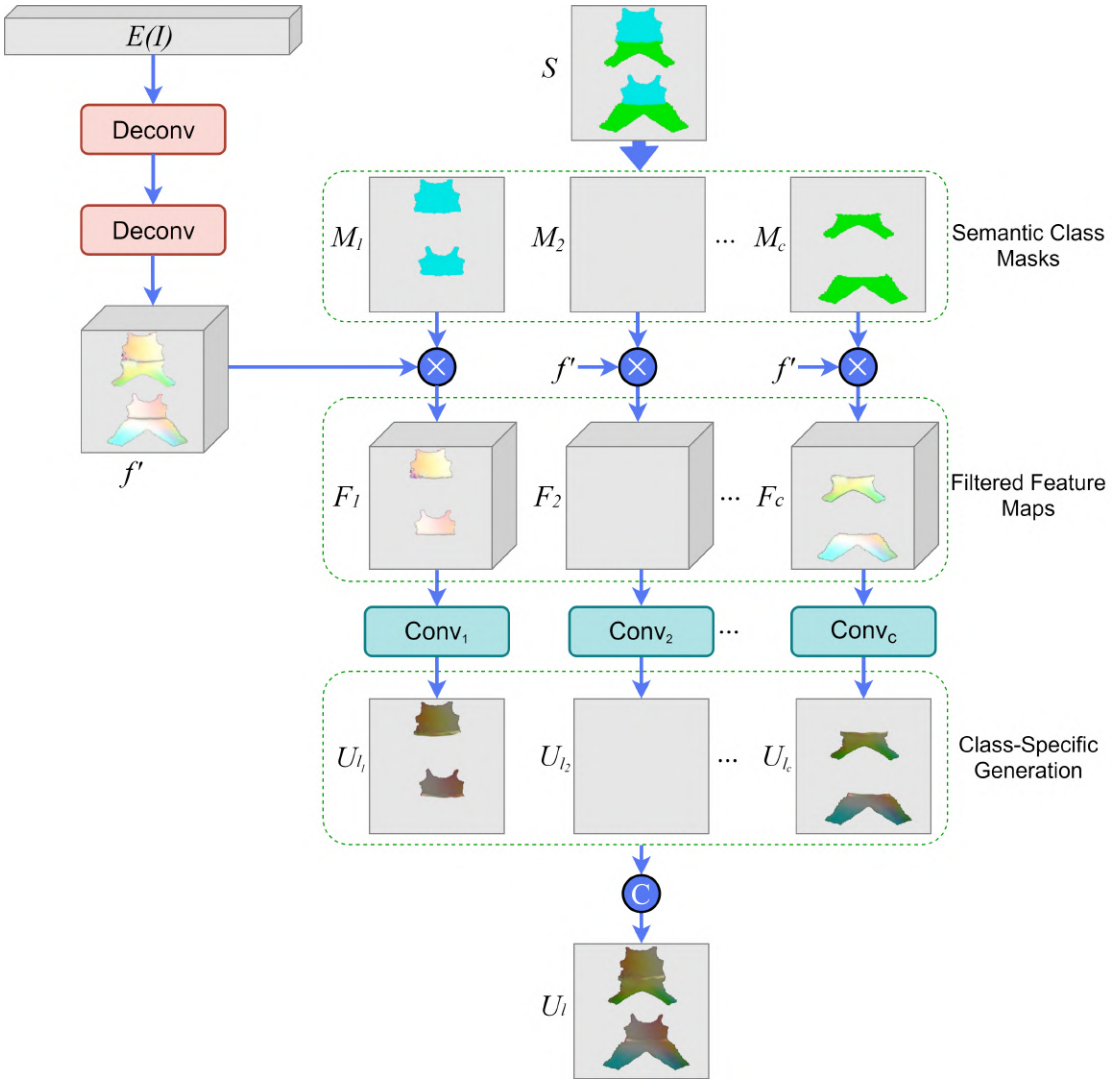


Figure 4.6: Class-Specific Local Generation Network overview. The symbol $\otimes$ denotes element-wise multiplication, and ⓒ channel-wise concatenation. Source: Own elaboration adapted from [56].

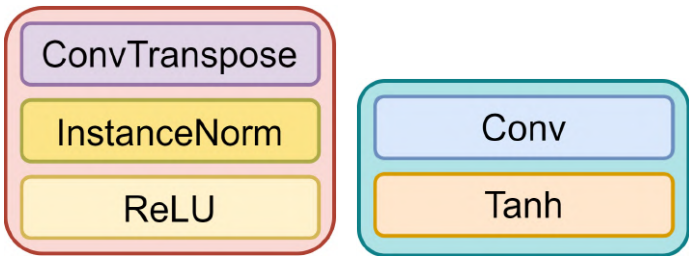
this step is a semantic-mask guided pixel-wise L1 reconstruction loss:

$$\mathcal{L}_{\text{L1}}^{\text{local}} = \sum_{i=1}^{c} \mathbb{E}_{U_r, U_{l_i}} \left[\| U_r \cdot M_i - U_{l_i} \|_1 \right] \tag{4.2}$$

Finally, to generate the final output U_l of the local generator, an element-wise addition of all the class-specific outputs is applied:

$$U_l^L = U_{l_1} \oplus U_{l_2} \oplus \cdots \oplus U_{l_c} \tag{4.3}$$

Class-Specific Discriminative Feature Learning To have more diverse generation for the different semantic classes, the authors also propose a novel classification-based feature learning module to learn more discriminative class-specific feature representations. Its main architecture is presented in Figure 4.8.



(a) Deconvolutional Block (b) Convolutional Block

Figure 4.7: Architecture of the deconvolutional and convolutional blocks. Source: Own elaboration.

This module receives as input a pack of feature maps produced from the different local generation branches $F_p = \{F_1, ..., F_c\}$. This packed feature map of dimension $c \times n \times h \times w$ (being n, h, w, the number of feature map channels, height and width, respectively) is fed into a semantic-guided averaging pooling layer, obtaining a pooled feature map of dimension $c \times n \times 1 \times 1$. Then, this pooled feature map is fed into a fully connected layer that predicts the classification probability of the c classes of the image. The output after this layer Y' has a dimension of $c \times c$, as for each filtered feature map F_i, the network predicts a $c \times 1$ one-hot vector with the probabilities of the c classes.

As in the input image do not appear all garment types/classes, features from local branches corresponding to the void classes should not contribute to the classification loss. For this reason, the loss defined here is a Cross-Entropy (CE) loss but filtering out the void classes by multiplying them with a void class indicator for each input sample. The indicator is a one hot vector $H = \{H_i\}_{i=1}^{c}$, with $H_i = 1$ for a valid class and $H_i = 0$ for a void one. The final CE loss is as follows:

$$\mathcal{L}_{\text{CE}} = -\sum_{m=1}^{c} H_m \sum_{i=1}^{c} 1\{Y(i) = i\} \log\left(f\left(F_i\right)\right) \tag{4.4}$$

being $1\{\cdot\}$ an indicator function that will return 1 if $Y(i) = i$, 0 otherwise, $f(\cdot)$ the function that produces the predicted classification probability given an input feature map $F(i)$, and Y the label set of all the classes.

Image-Level Global Generation Network Apart from the local generator, LGGAN contains a global generation network G_g that captures global structure information or layout of the target images, in our case, UV maps. As in the local generator, this module receives as input the encoded features $E(I)$. The global result U_g is obtained through a feed-forward computation $U_g = G_g\left(E\left(I\right)\right)$. G_g is formed by two deconvolutional blocks (Figure 4.7a) and one convolutional block (Figure 4.7b).

Pixel-Level Fusion Weight-Map Generation Network To combine local and global generated outputs, U_l and U_g, the LGGAN contains a pixel-level weight map generator G_w, which generates pixel-wise weights. This generator has the same structure than the global generator G_g, composed of two deconvolutional blocks

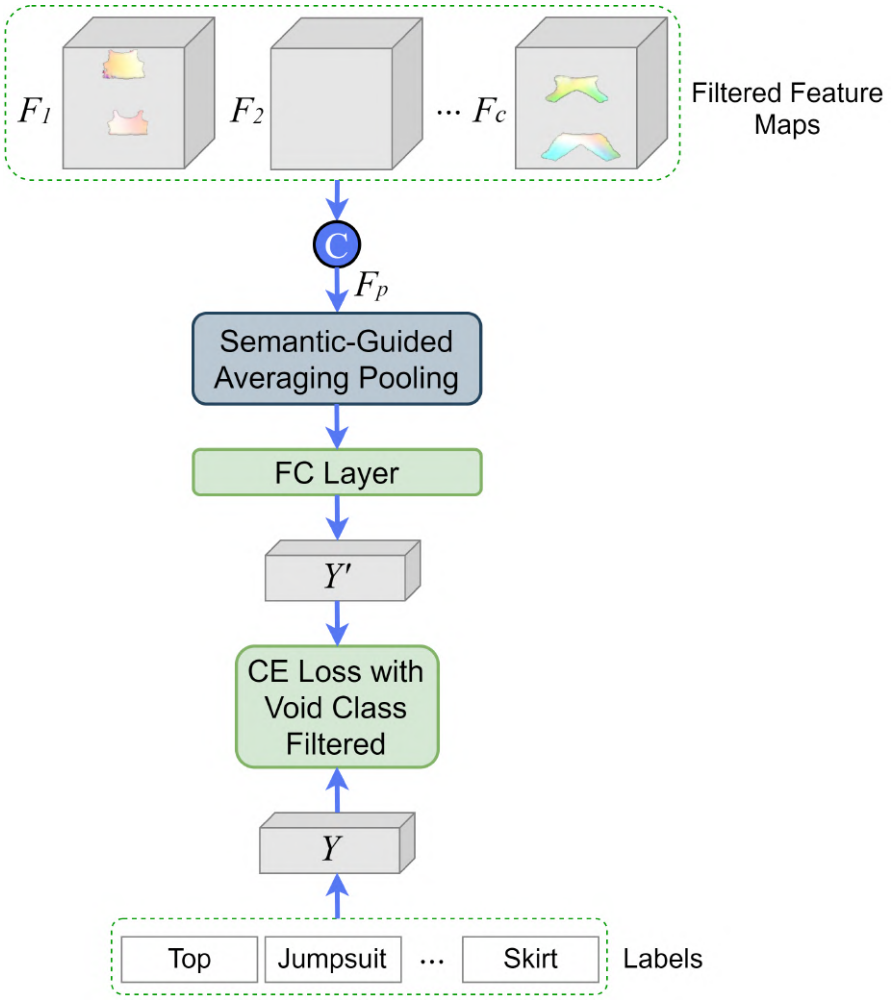


Figure 4.8: Class-Specific Discriminative Feature Learning architecture. Source: Own elaboration adapted from [56].

(Figure 4.7a) and one convolutional block (Figure 4.7b). However, in this case, the Tanh activation function of the convolutional block is replaced by a channel-wise softmax function, used for normalisation. Thus, the final output, a two-channel weight map W_f, is calculated as:

$$W_f = \text{Softmax}\left(G_w\left(E\left(S\right)\right)\right) \tag{4.5}$$

Finally, W_f is split to have a weight map W_l for the local generation and another one W_g for the global. The fused final generated UV map is computed as follows:

$$U_f = U_g \otimes W_g + U_l \otimes W_l \tag{4.6}$$

being $\otimes$ an element-wise multiplication operation.

4.2.1.2 Dual-Discriminator

The single domain vanilla discriminator from the original GAN [16] is extended to a cross domain structure named semantic-guided discriminator. Its inputs are the

semantic map S, the final output of the generator module (fake UV map) U_f and the ground truth output (real UV map) U_r.

$$\mathcal{L}_{\text{LGGAN}}(G, D) = \mathbb{E}_{S,U_r}\left[\log D\left(S, U_r\right)\right] + \mathbb{E}_{S,U_f}\left[\log\left(1 - D\left(S, U_f\right)\right)\right] \tag{4.7}$$

When the model is conditioned using the SMPL body of the person, the discriminator input is not the semantic guide S but the SMPL body UV map B. Therefore, the loss is modified as follows:

$$\mathcal{L}_{\text{LGGAN}}(G, D) = \mathbb{E}_{B,U_r}\left[\log D\left(B, U_r\right)\right] + \mathbb{E}_{B,U_f}\left[\log\left(1 - D\left(B, U_f\right)\right)\right] \tag{4.8}$$

4.2.2 3D loss functions

Apart from the losses of the original LGGAN model, described above, we add to our model four more types of 3D loss functions, which we apply on meshes. As most of these added losses come from works that use mesh-based representations, to apply them to our approach, we recover 3D meshes by unwrapping both generated and real UV maps in each step.

These losses are responsible for not only regressing the generated vertices to the ground truth, but also ensuring that the generated mesh converges to a smooth and uniform shape. More specifically, two of the four losses are responsible for the direct comparison between predicted and ground truth vertices and face normals ($\mathcal{L}_{\text{smoothL1}}$, $\mathcal{L}_{\text{normal}}$), while the other two are in charge of adding regularisation to prevent the network of getting stuck into some local minimum ($\mathcal{L}_{\text{laplacian}}$, $\mathcal{L}_{\text{edge}}$).

Below we detail the contribution of each of the losses and how they are computed. When formalising them, we use p for a vertex in the predicted mesh, q for a vertex in the ground truth mesh and $\mathcal{N}(p)$ for the set containing the neighbours of p.

Smooth L1 loss This loss is a combination of L1 and L2 losses, and we used it as a penalty to regress the predicted points p to its correct position, the ground truth q. It basically uses squared term (L2 term) if the absolute element-wise error falls below β and an L1 term otherwise. This makes it less sensitive to outliers and, in some cases, prevents exploding gradients [14].

$$\mathcal{L}_{\text{smoothL1}} = \begin{cases} 0.5\left(p - q\right)^2 / \beta, & \text{if } |p - q| < \beta \\ |p - q| - 0.5\beta, & \text{otherwise} \end{cases} \tag{4.9}$$

Surface normal loss This term forces the normal of the faces from the predicted mesh to be consistent with the ground truth normals. Coming from Pixel2Mesh work [61, 62], this loss is defined as:

$$\mathcal{L}_{\text{normal}} = \sum_{p} \sum_{q=\arg\min_q\left(\|p-q\|_2^2\right)} \left\|(p - k)^T \cdot n_q\right\|_2^2, \text{ s.t. } k \in \mathcal{N}(p) \tag{4.10}$$

being q the closest ground truth vertex to p found using chamfer distance, k a neighbour of p, and n_q the observed surface normal from ground truth at that vertex.

Laplacian smoothing regularisation This loss, from a work by Nelan *et al*. [41], prevents the vertices from moving too freely, avoiding the output mesh from deforming too much and ensuring that it has a smooth surface.

$$\mathcal{L}_{\text{laplacian}} = \sum_{p} \sum_{k \in \mathcal{N}(p)} \frac{1}{|\mathcal{N}(p)|}(k - p) \tag{4.11}$$

Edge length regularization To avoid having flying vertices, we penalise long edges by adding one last loss, extracted also from [61, 62].

$$\mathcal{L}_{\text{edge}} = \sum_{p} \sum_{k \in \mathcal{N}(p)} \|p - k\|_2^2 \tag{4.12}$$

Finally, the overall mesh loss is computed as a weighted sum of all these four losses, and contributes to all the three LGGAN generation branches, G_g, G_l and G_w, and the encoder E:

$$\mathcal{L}_{\text{mesh}} = \lambda_s \mathcal{L}_{\text{smoothL1}} + \lambda_n \mathcal{L}_{\text{normal}} + \lambda_l \mathcal{L}_{\text{laplacian}} + \lambda_e \mathcal{L}_{\text{edge}} \tag{4.13}$$

5. Experiments and Results

In this chapter, we present and analyse the results of the most important experiments and studies done in our work. We first present the concrete dataset used and the experimental setup used for training. Finally, we review the results obtained and discuss them quantitatively and qualitatively.

5.1 Dataset

The dataset used for training is CLOTH3D [4], which was introduced in 2020 as the first large-scale synthetic dataset of 3D clothed human sequences. It has over 2 million 3D samples with a large variety of garment type, topology, shape, size, tightness and fabric. Garments are simulated on top of thousands of different human pose sequences and body shapes, generating realistic cloth dynamics.

The fact the authors generate synthetic data, allows them to create a huge dataset that can be used in data-hungry deep learning approaches like ours, while having a large variability of examples. This is because synthetic data is much easier to generate than real data, usually coming from 3D scans, which are costly. Moreover, synthetic data has demonstrated to be suitable for training deep learning models to be used in real-life applications [48, 59].

Each sequence contains multiple frames of a human body in 3D with different poses. The human body is generated through SMPL [34] and animated through a valid sequence of SMPL pose parameters, taken from the work of [59] and having from around 2,600 sequences of 23 different actions (dancing, playing, running, walking, jumping, climbing, etc.). To generate the garment samples, authors automatically generate for each sequence garments with random shape, tightness, topology and fabric, resizing them to the target human shape and yielding a unique outfit for each sequence. In Figure 5.1 we can see an example of some frames belonging to a sequence of the dataset.

To be more specific, the dataset used is an extension of CLOTH3D, which includes the UV maps of the garments, that, as explained in previous chapters, are the 3D representation we use in this work. As the samples are layered, meaning each garment and body are represented by different 3D meshes and UV maps, when we have multiple garments in one sample we just join their UV maps in a single one. If there is overlap between the multiple garments, in the region where it occurs we keep just the upper-body garment vertices.

Due to the large number of samples in CLOTH3D dataset, the limited resources available and the time frame and scope of the master thesis, we use a subset of

Figure 5.1: Some examples of a sequence of the CLOTH3D dataset.

CLOTH3D to train and test our work. This subset contains about 2,000 non-overlapping sequences of 300 frames, resulting in a total of 600,000 samples. 1,396 of these sequences are used as training set, 139 as validation set and 438 as testing set, resulting in 418,800 training samples, 41,700 validation samples and 131,400 test samples. Therefore, even though we use a subset, this is large and varied enough to rely on the insights and conclusions of our experiments. Note that the data split is done by sequences, which ensures no garment, shape or pose is repeated in training and test.

The garment types included in the whole dataset and in our subset are the following: *Top*, *T-shirt*, *Trousers*, *Jumpsuit*, *Skirt* and *Dress*. An example of each garment type can be seen below, in Figure 5.2. All of these types vary in shape and topology, thus forming a wide variety of garment models. In Figure 5.3 we show different samples of *dresses*.

(a) Top (b) T-shirt (c) Trousers (d) Jumpsuit (e) Skirt (f) Dress

Figure 5.2: Examples of the different garment types in CLOTH3D dataset.

(a) (b) (c) (d) (e)

Figure 5.3: Examples of different dresses in CLOTH3D dataset.

Analysing the distribution of garment types in the CLOTH3D dataset, we observe that it is not balanced, *i.e.* there are more samples of some types than others. In our subset, it happens exactly the same thing. In Figure 5.4 we show the number

of sequences per type of garment in our three set splits. The types of clothes that appear in more sequences are *jumpsuits*, *dresses* and *trousers*, then *tops* and *T-shirts*, and finally *skirts*.

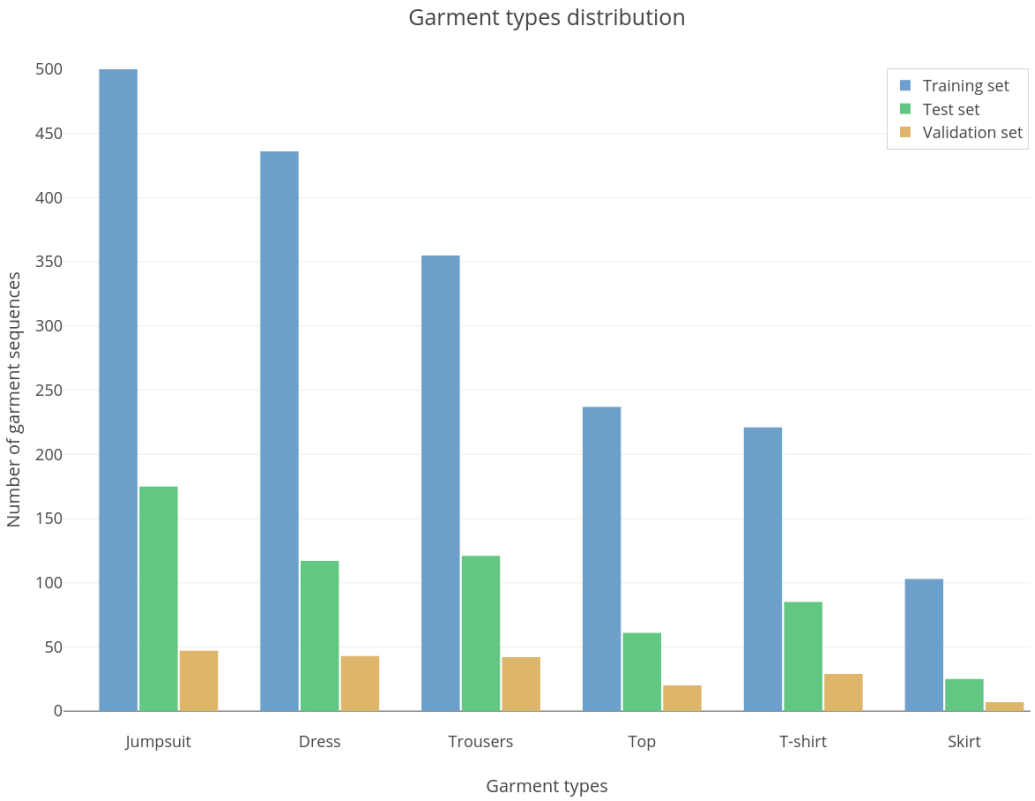


Figure 5.4: Garment type distribution of the CLOTH3D subset used in this work.

Nevertheless, having this imbalanced distribution does not affect the performance of our model, since its class-specific local generation network alleviates the influence of imbalanced training data.

5.2 Experimental setup

Before presenting our main experiments and their discussion, we explain the strategies followed to train our model and the hyperparameters fixed, as well as the metrics used to evaluate the different models.

5.2.1 Training details

RGB images pre-processing In each sequence of our dataset, people move through the 3D space doing actions, while the camera capturing the frames remains static in its location. For this reason, in different frames of the same sequence, the person can be seen in different positions, as they may have moved further to the right or to the left (with respect to the camera), or be seen in various sizes, as they may have moved away from or closer to the camera.

This is why we believe it is necessary to resize the image and crop/scale it so that all the input images of the dataset have the human centred in the middle and with the same proportion. To do so, we use the camera and human locations, and we define a bounding box equal for all frames and sequences, with the human centred on it. After that, all input images are resized to have the same size, 256×256, which is equal to the size of the UV maps. In Figure 5.5 we can see an example of an original frame of the dataset and its resized and scaled version.

(a) Original RGB image (b) Pre-processed RGB image

Figure 5.5: Example of an original image and its pre-processed version.

Parameter settings During the development of this work, we were able to extract some insights from our preliminary experiments, that led us to fix some parameters for the rest.

First of all, we defined the number of training epochs to 50, with a batch size of 4. The fact that we used a batch of this size is basically because, due to the size of our model and input data, and the computational resources available, we could not increase it any further. The number of epochs in which we started training was 100. However, we reduced it after observing that in every experiment the model started to perform always worse, even on training set, at around epochs 30-35. For this reason, we decided to establish the maximum number of epochs to 50, and start reducing the learning rate from epoch 25 onwards, to try to have a more stable training and avoid getting stuck in local minima. With this change, we managed to keep the model improving for more iterations.

Finally, we also set a dropout of 0.5 to avoid overfitting on the training dataset, and set the following weights to the 3D mesh losses defined in Section 4.2.2 (Equation 4.13): $\lambda_s = 1.0$, $\lambda_n = 0.01$, $\lambda_l = 0.5$ and $\lambda_e = 0.5$. These values were fixed after performing several experiments trying to find a good balance between these four losses and the losses of the original LGGAN model. Pixel2Mesh [62], work on which some of our losses are based, uses smaller values for the weights of the normal loss, the edge length regularisation and the laplacian regularisation. Nevertheless, in our tests we found that there were flying vertices causing long edges and a dull surface, reason why we increased the weights of all three losses, thus improving the quality of the reconstructions. Last but not least, the β parameter of the $\mathcal{L}_{\text{smoothL1}}$ loss is set to 1, as default (Equation 4.9).

Optimisation and weight initialisation Our LGGAN model is trained and optimised in an end-to-end fashion. We follow the optimisation method in [16] to optimise our LGGAN model, *i.e.* one gradient descent step on generators and discriminator alternately. We first train the encoder E and the three generators G_g, G_l and G_w with D fixed and then we train D with E, G_g, G_l and G_w fixed. The solver used is Adam [26] with momentum terms $\beta_1 = 0.5$ and $\beta_2 = 0.999$, as in the original LGGAN work [56]. The initial learning rate used for Adam is 0.0001, but, as explained before, from epoch 25 onwards, we start to decrease it by 4e−6. The original LGGAN was trained with a learning rate of 0.0002, but in our preliminary experiments we found that in our case it was too high, since the observed training loss curves were unstable. Finally, network weights are initialised with Xavier [15] strategy.

Environment The experiments were performed on an NVIDIA GeForce GTX 1080 Ti GPU, with 11GB of memory and CUDA Version 11.0.

5.2.2 Evaluation metrics

To evaluate our approach and the different experiments we use two metrics. Both metrics evaluate the results based on mesh format, so during evaluation we transform UV map representations into 3D meshes.

The first metric is the one used in the ChaLearn 3D+Texture garment reconstruction competition[1] (NeurIPS 2020), in which they also tried to solve the same image to 3D garment reconstruction task on the same dataset, CLOTH3D. This metric is called surface-to-surface (S2S) and it is an extension of the chamfer distance (CD), a common metric to compute similarity between two point clouds. For each point in each point cloud, CD finds the nearest point in the other point set, and sums the square of distance up. The reason they used the S2S metric is because CD does not take surface into account and may produce non-zero error for perfect estimations. In contrast, S2S computes the distance based on the nearest face rather than nearest vertex. Its formalisation is expressed as follows:

$$\mathrm{S2S}\left(S_1, S_2\right) = \frac{0.5}{N_1} \sum_{p \in S_1} \min_{f_q \in S_2} \mathrm{dist}\left(p, f_q\right) + \frac{0.5}{N_2} \sum_{q \in S_2} \min_{f_p \in S_1} \mathrm{dist}\left(q, f_p\right) \tag{5.1}$$

where p and q are 3D vertices and f_p and f_q triangulated vertices belonging to surfaces S_1 and S_2, respectively, N_1 and N_2 are the number of vertices of S_1 and S_2, respectively, and $\mathrm{dist}\left(p, f\right)$ is the distance between vertex p and face f.

The other metric used is the root-mean-square error (RMSE), which is one of the most commonly used measures for evaluating the quality of predictions. It shows how far these fall from true values using Euclidean distance. It is computed with the equation below and used it to measure the differences between predicted and ground truth 3D vertices.

$$\mathrm{RMSE}\left(S_1, S_2\right) = \sqrt{\frac{1}{N} \sum_{i=1}^{N} \left(P_i - Q_i\right)^2} \tag{5.2}$$

[1] http://chalearnlap.cvc.uab.es/challenge/40/description/

where P are the vertices of surface S_1, Q the vertices of surface S_2, and N the number of vertices of the surfaces S_1 and S_2 (both must have the same number).

5.3 Results

In this section, we show in detail the results of the most relevant experiments of our work and draw insights from them. In these experiments, we focus on studying the viability of UV maps for RGB to 3D garment reconstruction, analysing the pre-processing techniques proposed in Section 4.1. We also study how our extended version of the LGGAN model, originally intended for cross-view image translation task, performs on this task and the contribution of the 3D mesh losses added to it.

5.3.1 Normalisation techniques

As explained previously, UV maps contain 3D coordinates that store the vertices of the garment model on a flat surface. Therefore, the values of UV maps can be highly variable and do not have a fixed range. In fact, the values of one sample tend to vary significantly from the values of another. So much so that, in our first experiments, the model was unable to even overfit on the training dataset. Generated reconstructions were very far from the ground truth.
For this reason, we implemented two ways of normalising these UV maps: by performing min-max normalisation with the min and max values of the training set, and by using displacement UV maps, which represent garment vertices as an offset over the SMPL body vertices.

First approach, min-max normalisation, does not work very well as it fails to predict the position of the garment with respect to the body and fails to detect human movements. It also has a clear bias towards predicting the most common body shapes in the dataset, being unable to reconstruct all body types well. These drawbacks can be observed in Figure 5.6. The second normalisation proposal, displacement UV maps, is the better of the two. As we can see in Figure 5.6, the reconstruction of this model is very good and accurate, not having bias and being able to capture human body shape and movement.

By looking into the error metrics of both approaches on the test set, we can also clearly observe that using displacement UV maps is a much better option. The disadvantage this approach has, encountered in previous works [1, 2, 6], is that the model finds it more difficult to predict the types of garments that are not tight to the body, like *dresses* or *skirts*. The metric values observed are in line with this statement, as they are the garment types in which the error is higher, especially in *skirts*.
Note that, as observed in previous Table 5.1, RMSE error is always higher than S2S error. This is as expected and it is due to the fact that the S2S metric, and also the chamfer distance (CD) metric in which it is based, compute the distance between vertices by finding the nearest vertex or face in the other mesh. Instead, RMSE is the exact error between vertices since it compares vertex p_i with the corresponding vertex q_i of the other mesh.

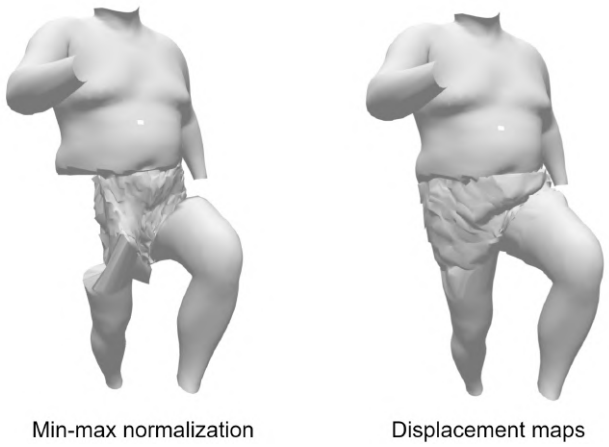


Figure 5.6: Examples of reconstructions of the different normalisation techniques.

Experiment	S2S error (in mm)						
	Top	T-shirt	Trousers	Jumpsuit	Skirt	Dress	All
Min-max normalisation	35.6	34.5	32.1	31.4	37.8	36.5	33.7
Displacement maps	8.8	10.9	10.0	8.5	24.1	15.9	11.4

Experiment	RMSE error (in mm)						
	Top	T-shirt	Trousers	Jumpsuit	Skirt	Dress	All
Min-max normalisation	77.3	93.3	81.9	81.4	99.3	98.3	87.1
Displacement maps	25.1	39.1	32.1	30.2	63.6	46.4	36.2

Table 5.1: Error comparison between the experiments using different normalisation techniques. Used S2S and RMSE errors (in mm) on test set. For both errors, small is better.

5.3.2 Inpainting UV maps

Inpainting of UV maps, *i.e.* filling the gaps between vertices to have a smoother UV map image, is another pre-processing technique introduced in this work. While there are many normalisation techniques used in the field of machine learning and 3D reconstruction, we are not aware that this technique has ever been used for this kind of task.

That is why we run some experiments with and without applying this pre-processing technique. In Figure 5.7 we can see the difference between applying or not this pre-processing step. The 3D reconstruction when UV maps do not have empty spaces in-between vertices is much smoother. The other generated mesh has far more wrinkles and noticeable protrusions.

Although the most notable difference appears to be the degree of smoothness, from the evaluation of the metrics in Table 5.2 we see that this technique also gives a significant boost to the model's performance, reducing its error by a few millimetres (mm).

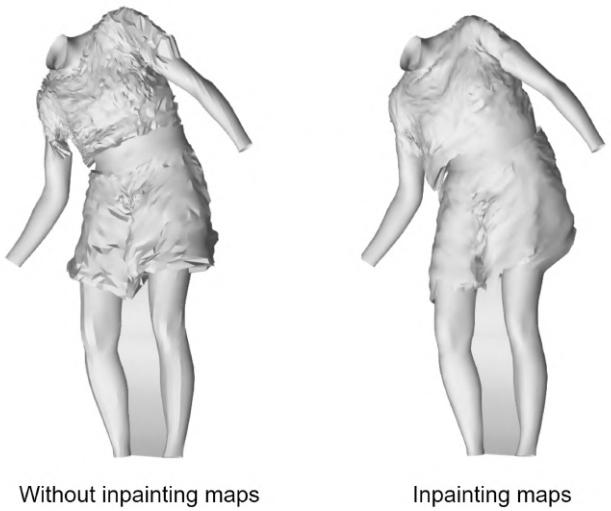


Figure 5.7: Example of a reconstruction when using the inpainting technique on UV maps and when not.

Experiment	S2S error (in mm)						
	Top	T-shirt	Trousers	Jumpsuit	Skirt	Dress	All
Without inpainting map	10.9	14.1	12.6	11.3	32.6	21.2	14.9
Inpainting map	8.8	10.9	10.0	8.5	24.1	15.9	11.4

Experiment	RMSE error (in mm)						
	Top	T-shirt	Trousers	Jumpsuit	Skirt	Dress	All
Without inpainting map	29.0	46.4	38.1	36.7	83.3	61.7	44.9
Inpainting map	25.1	39.1	32.1	30.2	63.6	46.4	36.2

Table 5.2: Error comparison between the experiment performing the inpainting pre-processing technique on UV maps and the one without doing it. Used S2S and RMSE errors (in mm) on test set. For both errors, small is better.

5.3.3 Semantic map vs Body UV map conditioning

As explained in Section 4.2, the LGGAN model allows adding a condition to guide and control the reconstruction. In our work, we make two proposals for conditioning it: using a semantic segmentation map of the target UV map to condition all generation branches and the discriminator, or using this semantic map to condition only the local generation branch and use the estimated SMPL body UV map of the person to condition the global generation branch and the discriminator. Figure 4.3 shows the LGGAN architecture overview on the first approach and Figure 4.4 shows it on the second.

Analysing the reconstructions of two executions, one conditioning on semantic maps and the other on the body UV map, we observe that there is no clear difference between the two options (Figure 5.8). Nevertheless, when taking a look at the evaluation results, presented in Table 5.3, we do see that conditioning on the body has a slightly lower error in all types of clothing. Even so, the difference is minimal,

so we consider both options to be good.

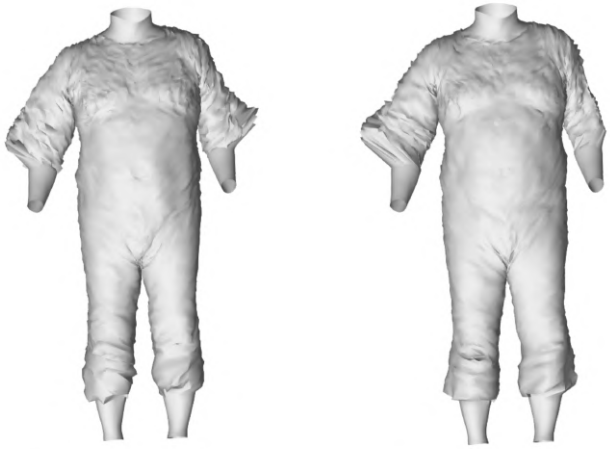


Figure 5.8: Examples of reconstructions of the different conditioning options.

Experiment	S2S error (in mm)						
	Top	T-shirt	Trousers	Jumpsuit	Skirt	Dress	All
Semantic segmentation	8.9	12.9	11.0	9.7	24.4	16.3	12.4
SMPL body UV map	8.8	10.9	10.0	8.5	24.1	15.9	11.4

Experiment	RMSE error (in mm)						
	Top	T-shirt	Trousers	Jumpsuit	Skirt	Dress	All
Semantic segmentation	24.8	42.1	33.2	32.1	62.7	46.9	37.5
SMPL body UV map	25.1	39.1	32.1	30.2	63.6	46.4	36.2

Table 5.3: Error comparison between the experiment conditioning on the semantic segmentation map and the one conditioning with the body SMPL body UV map. Used S2S and RMSE errors (in mm) on test set. For both errors, small is better.

5.3.4 Removing local generation branch

The LGGAN model is composed of three different branches: a semantic-guided class-specific generator modelling local context, an image-level generator modelling the global features, and a weight-map generator for fusing the local and the global generators.

In this experiment, we evaluate what happens if we leave only one of these three generation branches, the global generator. As we remove the local generation branch, in this case we do not condition on body UV map but on the semantic map, to still help the network with some class information.

As we can see in the reconstruction example below, in Figure 5.9, removing the local generation branch produces very bad results at the edges where the front and back of the garment model are joined, having a lot of flying vertices. The

quantitative evaluation, in Table 5.4, also shows that performance is significantly worsened by removing this local branch from the model and leaving only the global generator.

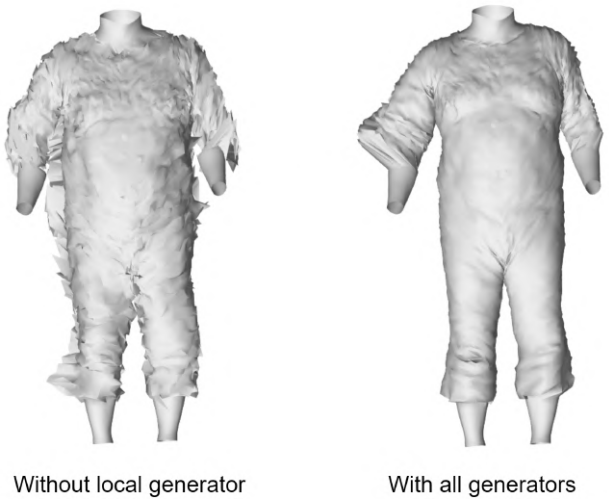


Figure 5.9: Example of a reconstruction of a model with the local generation branch and one without it.

Experiment	S2S error (in mm)						
	Top	T-shirt	Trousers	Jumpsuit	Skirt	Dress	All
Without local generator	14.2	15.4	14.9	14.0	35.7	26.3	17.6
Including all generators	8.8	10.9	10.0	8.5	24.1	15.9	11.4

Experiment	RMSE error (in mm)						
	Top	T-shirt	Trousers	Jumpsuit	Skirt	Dress	All
Without local generator	43.3	45.0	44.8	43.7	87.7	77.2	56.3
Including all generators	25.1	39.1	32.1	30.2	63.6	46.4	36.2

Table 5.4: Error comparison between the experiment with the local generation branch and the one without it. Used S2S and RMSE errors (in mm) on test set. For both errors, small is better.

These results make a lot of sense if we take a look at the attention weights generated by the fusion weight-map generator (Figure 5.10) when the local branch is kept in the model. The local weight map indicates that the local branch of the model is mostly paying attention to the edges of the UV maps. For this reason, when removing this branch, at the edges of the mesh where the front and back of the garments (separated in the UV map) are joined, these artefacts are produced.

5.3.5 3D mesh losses

During the development of the project and in the experiments carried out, we saw that the generated reconstructions were not very smooth, but had many lumps and irregularities compared to the ground truth. For this reason, we decided to

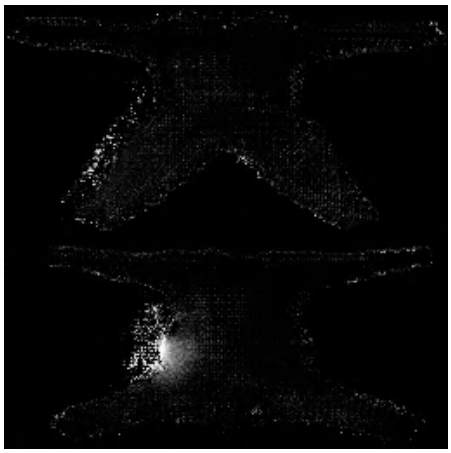


Figure 5.10: Example of an attention local weight map.

add some 3D loss functions applied on meshes, as explained in Section 4.2.2, to try to improve the quality of the reconstructions.

These losses are in charge of regressing the generated vertices to the ground truth, as well as ensuring that the generated mesh converges to a smooth and uniform shape, penalising flying vertices.

Therefore, the last experiment done is to analyse whether adding these 3D losses really improves the performance of our model or not. To do so, we train two models: one which computes and takes into account these losses, and another without it. Below, in Figure 5.11, we see the comparison between the reconstructions of one experiment and the other. We can observe that the meshes generated by the model with the 3D losses are much more uniform and therefore more realistic, having much fewer wrinkles.

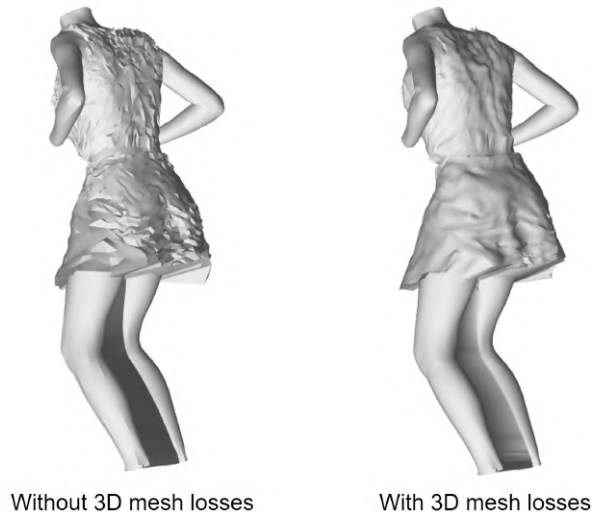


Figure 5.11: Example of a reconstruction of a model computing 3D mesh losses and one without doing it.

Despite this, if we evaluate the model in terms of the quantitative error between the

prediction and the ground truth, the difference is negligible, as seen in Table 5.5. In fact, for *skirts* and *dresses*, the error in the model without these losses is even smaller. As discussed in previous experiments, we know that our model already struggles with such loose-fitting garments, which have significant displacements over the body shape, because it represents garments as an offset over SMPL. This gets even worse when we add these 3D losses and penalise flying vertices, as the model has even less freedom to predict these large displacements. Still, as mentioned above, the difference between the errors of one model and the other is very small.

Experiment	S2S error (in mm)						
	Top	T-shirt	Trousers	Jumpsuit	Skirt	Dress	All
Without 3D mesh losses	9.1	11.6	10.1	9.1	23.5	15.9	11.7
With 3D mesh losses	8.8	10.9	10.0	8.5	24.1	15.9	11.4

Experiment	RMSE error (in mm)						
	Top	T-shirt	Trousers	Jumpsuit	Skirt	Dress	All
Without 3D mesh losses	29.0	46.4	38.1	36.7	83.3	61.7	44.9
With 3D mesh losses	25.1	39.1	32.1	30.2	63.6	46.4	36.2

Table 5.5: Error comparison between the experiment including mesh losses and the one without them. Used S2S and RMSE errors (in mm) on test set. For both errors, small is better.

5.4 Ablation study

In this section, we summarise the results and discussions presented before in the form of an ablation study, analysing the contribution of each different idea/technique designed and tested, in a qualitative and quantitative manner.

From our experiments, we conclude that our best model, from now on referred to as full model, is the one that uses displacement UV maps and applies the inpainting technique to them. It also conditions on SMPL body UV map and computes the mesh losses. In this ablation study, we take apart one component of our system at a time and show both S2S and RMSE error metrics on test set, as well as some 3D reconstruction examples.

5.4.1 Quantitative analysis

The quantitative analysis summary is shown in Table 5.6. As we can observe, the full model is the one with the best performance, having an average S2S error of 11.4 mm and an average RMSE error of 36.2 mm. The parts that seem to contribute less to the performance of the model are the mesh losses and the conditioning on the body UV map, since the errors remain practically unchanged.

On the other hand, the part that seems to contribute the most to the performance of the model is using displacement UV maps as our 3D representation. When we use the original UV maps, normalised using the min-max normalisation, the

performance in the test set decreases dramatically, having a much higher error in all types of clothing. It is true that, for *dresses* and *skirts*, this error does not increase to the same extent as in the rest of garment types.

The class-level local generator and the inpainting technique applied in the pre-processing of the UV maps are also key to our model, helping it to decrease its error significantly in most of the garment types. Of these two parts, the local generator is the one that has the greatest impact.

Model	S2S error (in mm)						
	Top	T-shirt	Trousers	Jumpsuit	Skirt	Dress	All
Full model	8.8	10.9	10.0	8.5	24.1	15.9	11.4
-Mesh losses	9.1	11.6	10.1	9.1	23.5	15.9	11.7
-Condition on body	8.9	12.9	11.0	9.7	24.4	16.3	12.4
-Displacement maps	35.6	34.5	32.1	31.4	37.8	36.5	33.7
-Local generator	14.2	15.4	14.9	14.0	35.7	26.3	17.6
-Inpainting	10.9	14.1	12.6	11.3	32.6	21.2	14.9

Model	RMSE error (in mm)						
	Top	T-shirt	Trousers	Jumpsuit	Skirt	Dress	All
Full model	25.1	39.1	32.1	30.2	63.6	46.4	36.2
-Mesh losses	26.0	40.9	33.0	32.1	63.5	46.1	37.5
-Condition on body	24.8	42.1	33.2	32.1	62.7	46.9	37.5
-Displacement maps	77.3	93.3	81.9	81.4	99.3	98.3	87.1
-Local generator	43.3	45.0	44.8	43.7	87.7	77.2	56.3
-Inpainting	29.0	46.4	38.1	36.7	83.3	61.7	44.9

Table 5.6: Ablation study that evaluates the contribution of the different ideas designed and tested to the performance of our model. We evaluate it computing S2S and RMSE errors (in mm) per garment type on test set. For both errors, small is better.

5.4.2 Qualitative analysis

To conclude the ablation study, in Figures 5.12, 5.13, 5.14, and 5.15 we show a qualitative analysis of the 3D reconstructions of different types of garment when removing one component of our final system at a time, as done above.

As we can see, the full model is the one that has more realism and has the best resemblance with the ground truth. Our model is capable of learning and modelling the dynamics of garments, regardless of the action, pose, race, gender and shape of the human. It is also able to accurately reconstruct garments regardless the image point of view, recovering the garment parts that are visible, as well as those that are occluded. However, the reconstructions of our best model are not perfect at all. It is still not capable of generating garments as smooth as those of the ground truth, and has difficulties on garments with large displacements over the body, as

can be seen in Figure 5.15, where the skirt of the prediction is much closer to the body than the one of the ground truth.

In this analysis, we can see a much bigger difference between the full model and the ones without mesh losses or without conditioning on the body, than the one we saw by looking at the evaluation metrics, as the difference was minimal. In both cases, the generated reconstructions are not as smooth as the reconstruction of the full model. This was expected for the model without the mesh losses, but it is surprising that it also happens when we condition only on the semantic map. Even so, as we have already said, the predicted garment global shape is still very good in both cases.

With these examples, we can confirm that the major contribution comes from using displacement UV maps. Except for one of the examples, shown in Figure 5.13, we see that the model not using displacement UV maps has difficulties in predicting the position of the garment with respect to the human and reconstructs it far away from the body. Also, we observe that, although in Figure 5.13 the shape and position of the garment may be acceptable, the model is not able to predict the leg's movement and reconstructs the garment as if the person were static.

Lastly, we observe that when not using the inpainting technique or removing the local generator from the LGGAN model, reconstructions have lots of lumps and wrinkles. Furthermore, the generated garments coming from the model without the local generation branch, have quite significant artefacts at the edges where the front and the back of the garment are joined.

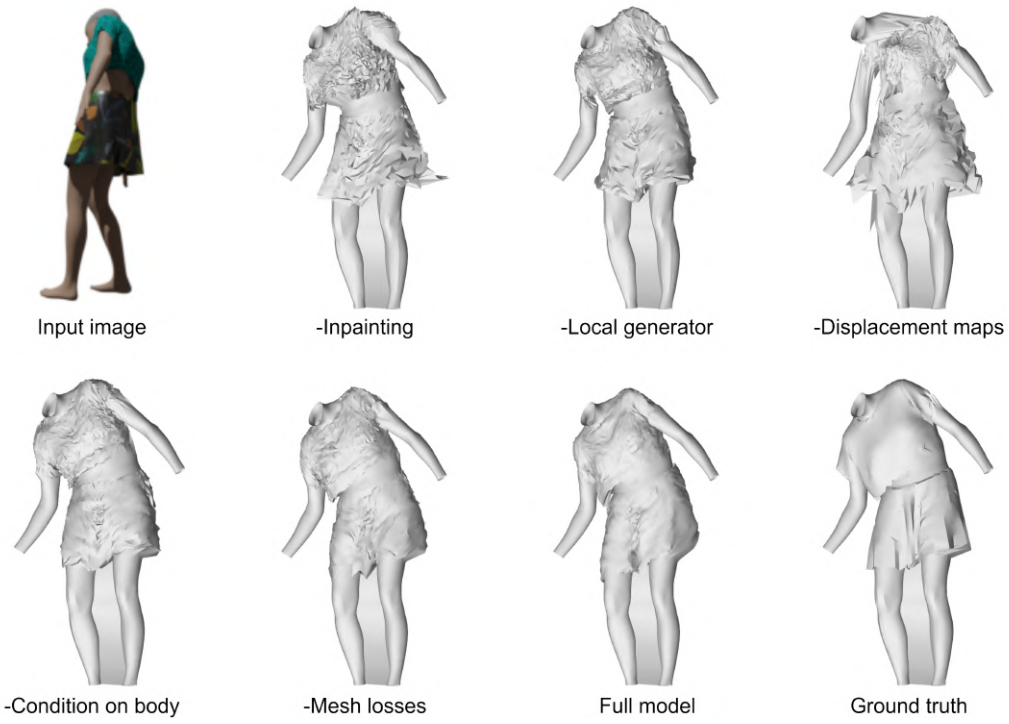


Figure 5.12: Ablation study on a T-shirt+Skirt sample of the test set

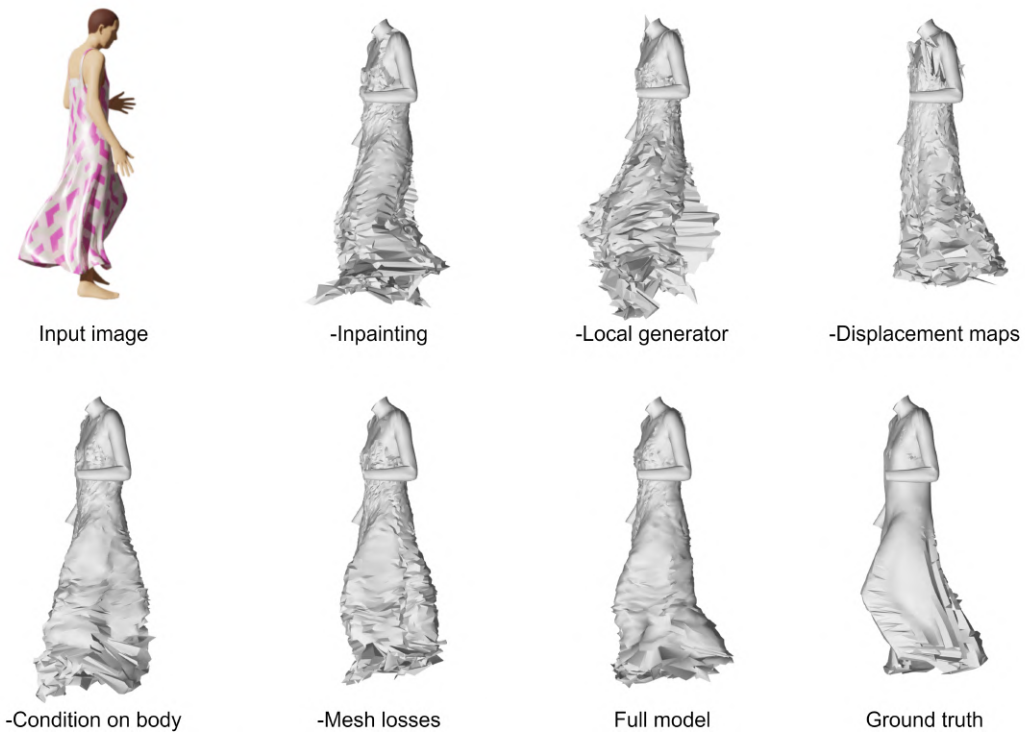


Figure 5.13: Ablation study on a Dress sample of the test set

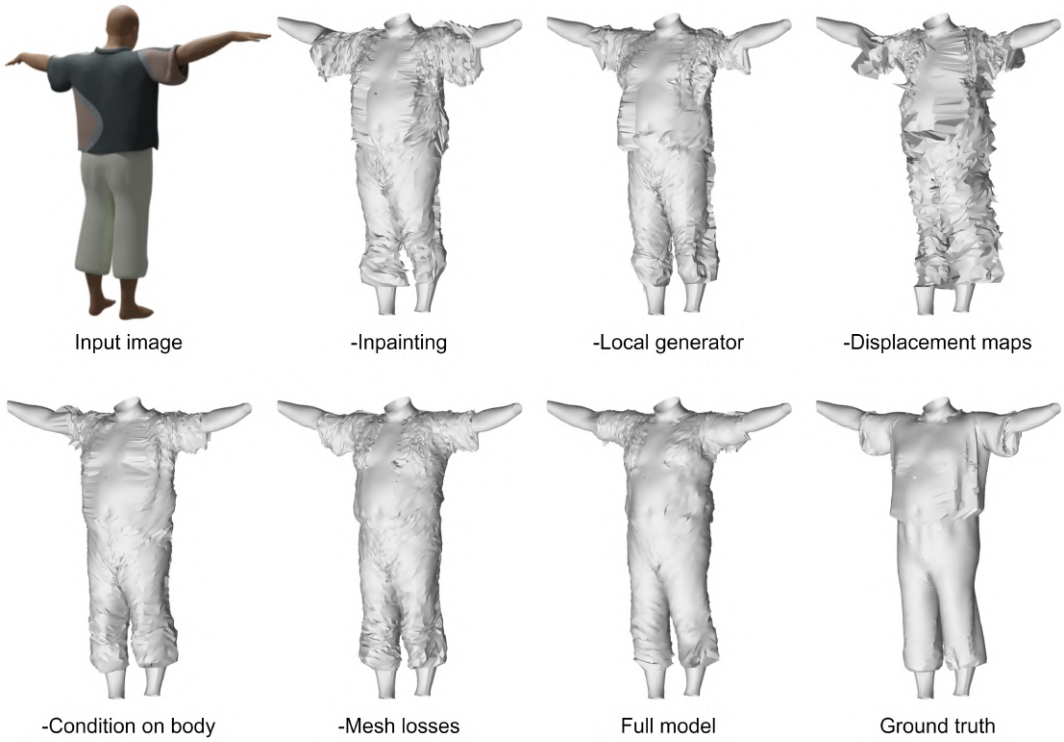


Figure 5.14: Ablation study on a T-shirt+Trousers sample of the test set

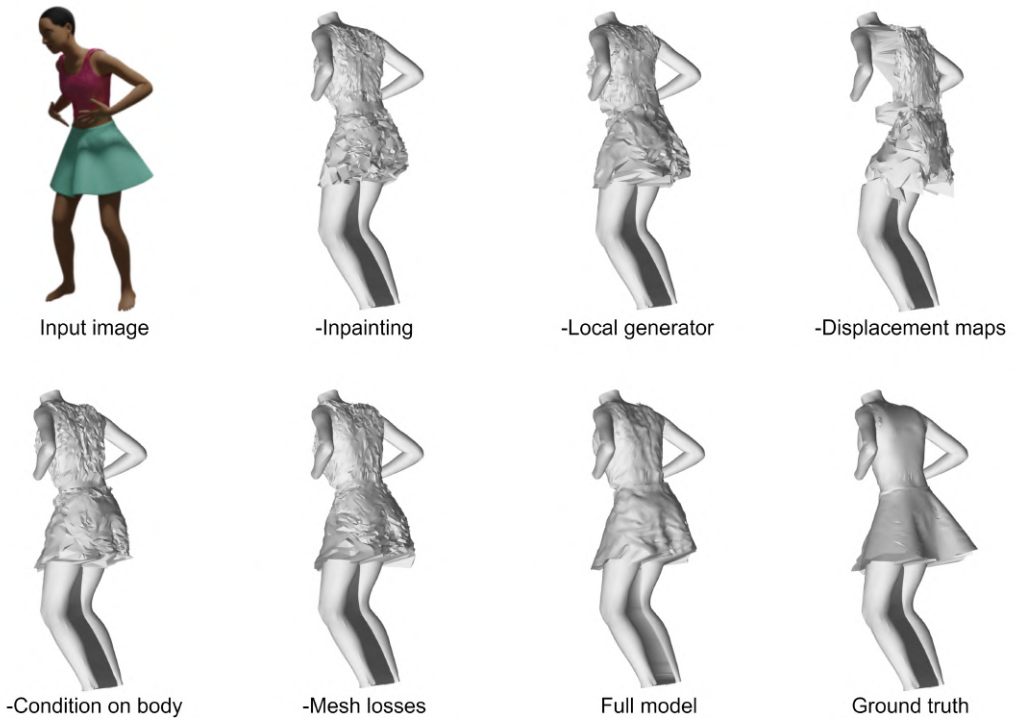


Figure 5.15: Ablation study on a Top+Skirt sample of the test set

5.5 Comparison with SOTA

After testing the different techniques and ideas proposed in this work, comparing all of them and analysing their contribution, we believe it is very interesting to compare also our model with some other state-of-the-art model in the field of 3D garment reconstruction from images.

To the best of our knowledge, the best current state-of-the-art model in the task of RGB to 3D garment reconstruction for the CLOTH3D [4] dataset is [35], a work accepted and in the process of publication. We compare our model against it in Table 5.7, using the S2S metric per garment type. It can be observed that our model obtains a lower error across all garment types, with an average error of about 20 mm less. The garment type in which we get the slightest difference is in *skirts*, in which we already know our model has the worst performance.

Model	S2S error (in mm)						
	Top	T-shirt	Trousers	Jumpsuit	Skirt	Dress	All
Madadi *et al.* [35]	23.9	43.3	40.6	22.1	32.8	29.3	31.3
Ours	8.8	10.9	10.0	8.5	24.1	15.9	11.4

Table 5.7: S2S error (in mm) per garment type comparison against SOTA in CLOTH3D dataset. Small is better.

6. Conclusions and future work

To sum up, in this thesis we have presented a system that learns garment dynamics and reconstructs 3D garment models from a single RGB image, by using UV maps to represent 3D data.

There are few precedents in the use of UV maps for 3D reconstruction tasks, so we believe that the study carried out in this work and the proposed pre-processing techniques can be of great value to the community.

The task of inferring the 3D geometry of garments with just a single image is not easy at all, since models must deal with lighting, occlusions, viewpoint, dynamism, etc. Nevertheless, our model achieves state-of-the-art results on the CLOTH3D dataset, generating good quality and realistic reconstructions. In this way, it demonstrates that it has the capacity to learn the dynamics of garments and infer their shape regardless of their topology and the human's action, pose, gender, race and body shape. It is also able to reconstruct both visible and occluded parts of garments accurately.

Furthermore, in this work, we have been able to reconstruct garments of different topology and type with just a single system, thus avoiding having to train a different network for each type as in other works. This is because our model allows the generation to be conditioned and controlled, so we use a semantic map to condition a local class-specific generation network that separately constructs a generator for each garment type.

Moreover, we have analysed the contribution of each of the proposals and techniques designed, learning the importance of applying the correct pre-processing techniques to the UV maps before starting the model training. We have also learned how a model, originally designed for a specific task, can be adapted for a completely different domain and still make it work.

As limitations, we have observed that our model has a much higher error than the average in dresses and skirts, and that the garment predictions are not as smooth as the real ones, having quite a few more wrinkles, especially when the poses of the human are strange.

As future work, we propose different lines of work that could be interesting:

- First, we suggest adding a pre-processing technique for the skirt and dress UV maps, to try to mitigate the problems we currently have in these two garment types.

- We also believe that it would be interesting to test the task of garment shape transition and editing in our model, by conditioning it on a different garment type and topology than the one in the input image, and see if it is able to change the garment of the human in the generated reconstruction, maintaining the same pose.
- Since the CLOTH3D dataset is composed of 3D clothed human sequences, a possible line of work could be to take advantage of the multiple frames in a unique sequence and learn the temporal data to improve the model's current performance.
- Finally, our last suggestion is to somehow predict not only the garment shape and topology but also the garment texture, which makes the task much more difficult.

Personally, I am very satisfied with the work done in this thesis, as I had never worked on a task involving 3D data before, and it has been a completely new world for me. Even so, applying many of the techniques and knowledge learned during the Master in Artificial Intelligence, I have developed a solution that has achieved the proposed objectives. In addition, during this development, I have learned many things that I did not know and that I believe will be of value for the future development of my career.

Bibliography

[1] Thiemo Alldieck, Marcus Magnor, Bharat Lal Bhatnagar, Christian Theobalt, and Gerard Pons-Moll. Learning to reconstruct people in clothing from a single rgb camera, 2019.

[2] Thiemo Alldieck, Gerard Pons-Moll, Christian Theobalt, and Marcus Magnor. Tex2shape: Detailed full human body geometry from a single image, 2019.

[3] M. Bertalmio, Andrea Bertozzi, and G. Sapiro. Navier-stokes, fluid dynamics, and image and video inpainting. volume 1, pages I–355, 02 2001.

[4] Hugo Bertiche, Meysam Madadi, and Sergio Escalera. Cloth3d: Clothed 3d humans, 2020.

[5] D.N. Bhat and S.K. Nayar. Ordinal Measures for Image Correspondence. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(4):415–423, Apr 1998.

[6] Bharat Lal Bhatnagar, Garvita Tiwari, Christian Theobalt, and Gerard Pons-Moll. Multi-garment net: Learning to dress 3d people from images, 2019.

[7] Federica Bogo, Angjoo Kanazawa, Christoph Lassner, Peter Gehler, Javier Romero, and Michael J. Black. Keep it smpl: Automatic estimation of 3d human pose and shape from a single image, 2016.

[8] Derek Bradley, Tiberiu Popa, Alla Sheffer, Wolfgang Heidrich, and Tamy Boubekeur. Markerless garment capture. *ACM Trans. Graphics (Proc. SIGGRAPH)*, 27(3):99, 2008.

[9] Cesar Cadena, Luca Carlone, Henry Carrillo, Yasir Latif, Davide Scaramuzza, Jose Neira, Ian Reid, and John J. Leonard. Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age. *IEEE Transactions on Robotics*, 32(6):1309–1332, Dec 2016.

[10] Angel X. Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, Jianxiong Xiao, Li Yi, and Fisher Yu. Shapenet: An information-rich 3d model repository, 2015.

[11] Christopher B. Choy, Danfei Xu, JunYoung Gwak, Kevin Chen, and Silvio Savarese. 3d-r2n2: A unified approach for single and multi-view 3d object reconstruction, 2016.

[12] Enric Corona, Albert Pumarola, Guillem Alenyà, Gerard Pons-Moll, and Francesc Moreno-Noguer. Smplicit: Topology-aware generative model for clothed people, 2021.

[13] Haoqiang Fan, Hao Su, and Leonidas Guibas. A point set generation network for 3d object reconstruction from a single image, 2016.

[14] Ross Girshick. Fast r-cnn, 2015.

[15] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *AISTATS*, 2010.

[16] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks, 2014.

[17] Thibault Groueix, Matthew Fisher, Vladimir G. Kim, Bryan C. Russell, and Mathieu Aubry. Atlasnet: A papier-mâché approach to learning 3d surface generation, 2018.

[18] Rıza Alp Güler, Natalia Neverova, and Iasonas Kokkinos. Densepose: Dense human pose estimation in the wild, 2018.

[19] Rana Hanocka, Amir Hertz, Noa Fish, Raja Giryes, Shachar Fleishman, and Daniel Cohen-Or. Meshcnn. *ACM Transactions on Graphics*, 38(4):1–12, Jul 2019.

[20] Richard Hartley and Andrew Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2 edition, 2004.

[21] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.

[22] Derek Hoiem, Alexei A. Efros, and Martial Hebert. Recovering surface layout from an image. *Int. J. Comput. Vision*, 75(1):151–172, October 2007.

[23] Zeng Huang, Yuanlu Xu, Christoph Lassner, Hao Li, and Tony Tung. Arch: Animatable reconstruction of clothed humans, 2020.

[24] Moon-Hwan Jeong, Dong-Hoon Han, and Hyeong-Seok Ko. Garment capture from a photograph. *Comput. Animat. Virtual Worlds*, 26(3–4):291–300, May 2015.

[25] Angjoo Kanazawa, Michael J. Black, David W. Jacobs, and Jitendra Malik. End-to-end recovery of human shape and pose, 2018.

[26] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017.

[27] Diederik P Kingma and Max Welling. Auto-encoding variational bayes, 2014.

[28] Nikos Kolotouros, Georgios Pavlakos, Michael J. Black, and Kostas Daniilidis. Learning to reconstruct 3d human pose and shape via model-fitting in the loop, 2019.

[29] Nikos Kolotouros, Georgios Pavlakos, and Kostas Daniilidis. Convolutional mesh regression for single-image human shape reconstruction, 2019.

[30] Anders Boesen Lindbo Larsen, Søren Kaae Sønderby, Hugo Larochelle, and Ole Winther. Autoencoding beyond pixels using a learned similarity metric, 2016.

[31] Aldo Laurentini. The visual hull concept for silhouette-based image understanding. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 16:150–162, 03 1994.

[32] Yann LeCun and Corinna Cortes. MNIST handwritten digit database. 2010.

[33] Yangyan Li, Rui Bu, Mingchao Sun, Wei Wu, Xinhan Di, and Baoquan Chen. Pointcnn: Convolution on $\mathcal{X}$-transformed points, 2018.

[34] Matthew Loper, Naureen Mahmood, Javier Romero, Gerard Pons-Moll, and Michael J. Black. SMPL: A skinned multi-person linear model. *ACM Trans. Graphics (Proc. SIGGRAPH Asia)*, 34(6):248:1–248:16, October 2015.

[35] Meysam Madadi, Hugo Bertiche, Wafa Bouzouita, Isabelle Guyon, and Sergio Escalera. Learning cloth dynamics: 3d + texture garment reconstruction benchmark, 2021.

[36] Meysam Madadi, Hugo Bertiche, and Sergio Escalera. Smplr: Deep smpl reverse for 3d human pose and shape recovery, 2019.

[37] Deepti Maduskar and Nitant Dube. Navier–stokes-based image inpainting for restoration of missing data due to clouds. In Manoj Kumar Sharma, Vijaypal Singh Dhaka, Thinagaran Perumal, Nilanjan Dey, and João Manuel R. S. Tavares, editors, *Innovations in Computational Intelligence and Computer Vision*, pages 497–505, Singapore, 2021. Springer Singapore.

[38] Priyanka Mandikal and R. Venkatesh Babu. Dense 3d point cloud reconstruction using a deep pyramid network, 2019.

[39] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets, 2014.

[40] Ryota Natsume, Shunsuke Saito, Zeng Huang, Weikai Chen, Chongyang Ma, Hao Li, and Shigeo Morishima. Siclope: Silhouette-based clothed people, 2019.

[41] Andrew Nealen, Takeo Igarashi, Olga Sorkine, and Marc Alexa. Laplacian mesh optimization. GRAPHITE '06, page 381–389, New York, NY, USA, 2006. Association for Computing Machinery.

[42] Mohamed Omran, Christoph Lassner, Gerard Pons-Moll, Peter V. Gehler, and Bernt Schiele. Neural body fitting: Unifying deep learning and model-based human pose and shape estimation, 2018.

[43] Georgios Pavlakos, Vasileios Choutas, Nima Ghorbani, Timo Bolkart, Ahmed A. A. Osman, Dimitrios Tzionas, and Michael J. Black. Expressive body capture: 3d hands, face, and body from a single image, 2019.

[44] Charles R. Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation, 2017.

[45] Charles R. Qi, Li Yi, Hao Su, and Leonidas J. Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space, 2017.

[46] Krishna Regmi and Ali Borji. Cross-view image synthesis using conditional gans, 2018.

[47] Gernot Riegler, Ali Osman Ulusoy, and Andreas Geiger. Octnet: Learning deep 3d representations at high resolutions, 2017.

[48] German Ros, Laura Sellart, Joanna Materzynska, David Vázquez, and Antonio López. The synthia dataset: A large collection of synthetic images for semantic segmentation of urban scenes. pages 3234–3243, 06 2016.

[49] Shunsuke Saito, Zeng Huang, Ryota Natsume, Shigeo Morishima, Angjoo Kanazawa, and Hao Li. Pifu: Pixel-aligned implicit function for high-resolution clothed human digitization, 2019.

[50] Shunsuke Saito, Tomas Simon, Jason Saragih, and Hanbyul Joo. Pifuhd: Multi-level pixel-aligned implicit function for high-resolution 3d human digitization, 2020.

[51] Philip Saponaro, Scott Sorensen, Stephen Rhein, Andrew R. Mahoney, and Chandra Kambhamettu. Reconstruction of textureless regions using structure from motion and image-based interpolation. In *2014 IEEE International Conference on Image Processing (ICIP)*, pages 1847–1851, 2014.

[52] Ashutosh Saxena, Min Sun, and Andrew Y. Ng. Make3d: Learning 3d scene structure from a single still image. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(5):824–840, 2009.

[53] Johannes L. Schonberger and Jan-Michael Frahm. Structure-from-motion revisited. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[54] David Smith, Matthew Loper, Xiaochen Hu, Paris Mavroidis, and Javier Romero. Facsimile: Fast and accurate scans from an image in less than a second, 2019.

[55] Zhaoqi Su, Tao Yu, Yangang Wang, Yipeng Li, and Yebin Liu. Deepcloth: Neural garment representation for shape and style editing, 2020.

[56] Hao Tang, Dan Xu, Yan Yan, Philip H. S. Torr, and Nicu Sebe. Local class-specific and global image-level generative adversarial networks for semantic-guided scene generation, 2020.

[57] Maxim Tatarchenko, Alexey Dosovitskiy, and Thomas Brox. Octree generating networks: Efficient convolutional architectures for high-resolution 3d outputs, 2017.

[58] Uv mapping. Uv mapping — Wikipedia, the free encyclopedia, 2021.

[59] Gul Varol, Javier Romero, Xavier Martin, Naureen Mahmood, Michael J. Black, Ivan Laptev, and Cordelia Schmid. Learning from synthetic humans. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jul 2017.

[60] Gül Varol, Duygu Ceylan, Bryan Russell, Jimei Yang, Ersin Yumer, Ivan Laptev, and Cordelia Schmid. Bodynet: Volumetric inference of 3d human body shapes, 2018.

[61] Nanyang Wang, Yinda Zhang, Zhuwen Li, Yanwei Fu, Wei Liu, and Yu-Gang Jiang. Pixel2mesh: Generating 3d mesh models from single rgb images, 2018.

[62] Chao Wen, Yinda Zhang, Zhuwen Li, and Yanwei Fu. Pixel2mesh++: Multi-view 3d mesh generation via deformation, 2019.

[63] Jiajun Wu, Chengkai Zhang, Tianfan Xue, William T. Freeman, and Joshua B. Tenenbaum. Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling, 2017.

[64] Yun-Peng Xiao, Yu-Kun Lai, Fang-Lue Zhang, Chunpeng Li, and Lin Gao. A survey on deep geometry learning: From a representation perspective, 2020.

[65] Haozhe Xie, Hongxun Yao, Xiaoshuai Sun, Shangchen Zhou, and Shengping Zhang. Pix2vox: Context-aware 3d reconstruction from single and multi-view images. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, Oct 2019.

[66] Xiangyu Xu, Hao Chen, Francesc Moreno-Noguer, Laszlo A. Jeni, and Fernando De la Torre. 3d human shape and pose from a single low-resolution image with self-supervised learning, 2020.

[67] Zerong Zheng, Tao Yu, Yixuan Wei, Qionghai Dai, and Yebin Liu. Deephuman: 3d human reconstruction from a single image, 2019.

[68] B. Zhou, Xiaowu Chen, Qiang Fu, K. Guo, and P. Tan. Garment modeling from a single image. *Comput. Graph. Forum*, 32:85–91, 2013.

[69] Heming Zhu, Yu Cao, Hang Jin, Weikai Chen, Dong Du, Zhangye Wang, Shuguang Cui, and Xiaoguang Han. Deep fashion3d: A dataset and benchmark for 3d garment reconstruction from single images, 2020.

[70] Cihan Öngün and Alptekin Temizel. Paired 3d model generation with conditional generative adversarial networks, 2019.