DOCTORAL THESIS

Towards Efficient and Realistic Animation of 3D Garments with Deep Learning

Autor: Hugo BERTICHE ARGILA

Director: Prof. Sergio ESCALERA

Co-Director: Dr. Meysam MADADI

UNIVERSITAT DE BARCELONA

Towards Efficient and Realistic Animation of 3D Garments with Deep Learning

Memòria presentada per optar al grau de doctor per la Universitat de Barcelona

Programa de doctorat en Matemàtiques i Informàtica

Autor: Hugo BERTICHE ARGILA

Director: Prof. Sergio ESCALERA

Co-Director: Dr. Meysam MADADI

Tutor: Prof. Sergio ESCALERA

UNIVERSITAT DE BARCELONA

Declaration of Authorship

I, Hugo BERTICHE ARGILA, declare that this thesis titled, "Towards Efficient and Realistic Animation of 3D Garments with Deep Learning" and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.

Signed:

Date:

UNIVERSITAT DE BARCELONA

Abstract

Facultat de Matemàtiques i Informàtica
Department de Matemàtiques i Informàtica

Doctor of Mathematics and Computer Science

Towards Efficient and Realistic Animation of 3D Garments with Deep Learning

by Hugo BERTICHE ARGILA

Machine learning has experienced a soar thanks to the proliferation of deep learning based methodologies. 3D vision is one of the many fields that benefited from this trend. Within this domain, I focused my research on human-centric scenarios. As a starting point, I begin with a 3D human pose and shape reconstruction approach from still images. Relying on a powerful CNN and a novel inverse graphics solution, I define the steps to predict volumetric humans as 3D meshes. As a natural extension, I turn my attention to the modelling of 3D garments for complete human representation. Deep learning models require huge volumes of data. For this reason, next, I explain my work developing the biggest 3D garment dataset, CLOTH3D. This was motivated by the lack of such data for the study of cloth on humans. Additionally, in the same context, I describe a baseline model for 3D garment generation trained on CLOTH3D. After identification of the major drawbacks of the baseline model, I introduce a novel solution for the garment animation problem. Deep learning models usually require data with a fixed dimensionality. Related works proposed expensive data pre-processings to make data uniform, albeit diminishing the quality, among other issues. By focusing purely in garment animation, I designed a fully-convolutional model that does not suffer from the aforementioned problem. This new model can animate even completely unseen outfits. Nonetheless, cloth animation is a tremendously complex problem. In practice, deep models which encode multiple garments end up showing poor quality. Moreover, I noted significant drawbacks in supervised learning schemes for garments. Motivated by these observations, I devised a novel technique that allows unsupervised training for the 3D garment animation task. As a consequence, this methodology leads to smaller, more robust models that can be obtained in a matter of minutes. Furthermore, it shows an unprecedented level of performance. Because of this, it became the first viable option for deep-based real-time garment animation in real life applications. Nonetheless, it is a quasi-static approach. Cloth dynamics are crucial for proper garment animation. Finally, the last of my contributions describes how to learn cloth dynamics unsupervisedly, making the solution for garment animation complete. Additionally, I establish the bases of this new unsupervised neural garment animation framework.

Acknowledgements

I would like to express my gratitude to my thesis advisor Professor Sergio Escalera, whose never-ending enthusiasm and immense support kept me strongly motivated through my whole Ph.D. I want to also thank my thesis co-advisor, Doctor Meysam Madadi, who has been there for guidance and always sparked interesting scientific discussion.

I thank as well the Institut de Matemàtiques, Universitat de Barcelona (IMUB) for giving me the opportunity to course my Ph.D. with a scholarship, the APIF.

Finally, I would also like to show my appreciation to the private research centers that allowed me to test my research skills in the field as an intern, Disney Research Zurich and Adobe Research.

Contents

To my beloved wife, Sonia, who has been there since before the very beginning, through all the ups and downs, rights and wrongs, and deserves this Ph.D. as much as I do.
To my cats, Arya and Zelda, who have always purred away and scratched off the burden and distress of the Ph.D.
To my parents, Miguel Ángel and Maria Eugenia, for their invaluable efforts in rising my siblings and me.
To all my siblings, Carlota, Alex, Borja, Iván and Luca, who gave their youngest brother the best childhood.
To my nieces, Cloe and Mia, who have the brightest smiles, the kindest eyes and the warmest hearts.

Chapter 1

Introduction

History of clothing dates back to prehistoric times. Modern studies suggest the first clothes were being worn as early as 170000 years ago, or as late as 90000 years ago (Barber, 1991; Barber, 1995). We cannot find in the world any other animal with the ability to gather, craft or wear clothes. It is an exclusively human characteristic. Furthermore, all human civilizations have developed this unique feature. At first, primarily functional fabrics were crafted from animal skin or vegetation as a mean to protect ourselves from the inclemency of the weather. This might have been a key factor in the success of humans in proliferating to other regions with harsher climate. Soon, mankind knowledge would improve into developing sewing tools and, later, into the crafting of textiles, as opposed to attaching pieces of animal skin together. Fig. 1.1 shows ancient samples of human weaving skills. From a piece of fabric as old as 34000 years to different garments few thousands of years old. Not much further in time, clothing would already serve a higher purpose within the cultural aspects of civilizations. Also playing an important role in ancient trading and as a symbol of social status. Fast-forwarding to the current age, the textile industry maintains an important position in our modern society, culture and economy. In a rapidly evolving technological world, humankind has given birth to a parallel digital reality. Hastily, the scientific community took an interest on transferring clothes into virtual worlds. The initial attempts, just a few decades ago, rely on the knowledge about geometry and the physical laws that govern cloth behaviour (Weil, 1986; Feynman, 1986; Terzopoulos et al., 1987). Since then, we have witnessed an exponential growth in the quality of 3D garment animation through Physically Based Simulation (Baraff et al., 1998; Carignan et al., 1992; Haumann, 1987; Breen et al., 1992; Provot et al., 1995; Narain et al., 2012; Pfaff et al., 2014; Kaldor et al., 2008; Kaldor et al., 2010). We find plenty of evidence about this in all major animation studio films, where it is far

FIGURE 1.1: Oldest preserved garments in the world. At left, the oldest samples of cloth and thread found, thought to be more than 34000 years old. In the middle, the Tarkhan Dress, which dates back to the end of the fourth millennium B.C. At right, an ancient pair of trousers that are approximately 3300 years old.

from uncommon to observe astoundingly realistic clothes. Nonetheless, such results come at a high computational price, while there still are plenty of applications that demand real-time garment animation. The main ones being video-games and Augmented/Virtual Reality (AR/VR). To this end, many recent efforts from the research community aimed to the implementation of faster, more efficient, cloth simulation algorithms. While significant progress has been achieved (Müller et al., 2007; Liu et al., 2013; Macklin et al., 2016) often quality is sacrificed or expensive hardware is required. This is not enough to bring faithful garment animation to scenarios where computing power is scarce (mobile devices) or is required for other complex tasks (video-games and AR/VR). Inspired by the recent success of deep learning in many 3D problems (Socher et al., 2012; Richardson et al., 2016; Qi et al., 2017; Han et al., 2017; Arsalan Soltani et al., 2017; Omran et al., 2018; Madadi et al., 2020), current trends explore the possibility of obtaining the coveted fast, efficient and realistic garment animation through neural networks. In this book, I will describe my journey into the scientific meadow of deep learning to the boundaries of the neural garment animation and how I pushed them further in a slight, but meaningful way.

All stories need a beginning, and for me, this was human 3D pose and shape estimation from RGB images. Classical computer vision methodologies were built around hand-crafted features. Then, with the appearance and success of deep learning methodologies, neural networks quickly became the standard. The problem of human pose and shape recovery was not an exception. In Chapter 2 I describe a deep-based solution for this problem inspired by the limitations seen in related works. Thanks to the combined efforts of the scientific community, human pose estimation has seen a significant progress since it was first tackled. In spite of that, even with accurate 3D shape recovery to complement pose, something important was missing. That is, cloth. As previously explained, clothes play an important role in our society. Human analysis, retrieval and modelling would not be complete without clothing. Deep learning has proved to be capable of solving an innumerable variety of tasks. Therefore, its application to cloth-related problems was the next natural step. Neural networks are data-hungry models, but public clothing dataset were scarce. In Chapter 3 I address this issue by defining, designing and generating the largest dataset of dressed humans, CLOTH3D. This dataset fills the gaps observed in related works: garment dynamics and outfit variability. While originally thought as a milestone towards dressed human reconstruction from images, due to its complexity, purely 3D clothing presented itself as an enthralling research subject. Along with the dataset, I also introduce a baseline generative model for 3D clothing. It was then clear that 3D garment generation and animation were two separate problems that needed different solutions. In Chapter 4 I define an approach for the specific problem of garment animation. One of the major challenges of working with 3D garments is the heterogeneity of their representations. Usually as 3D meshes, garments show a notable variability in terms of geometry, topology, mesh resolution, vertex order and connectivity. While neural networks usually require fixed input dimensionality and order. I propose a model architecture able to handle any kind of 3D mesh, allowing modelling an indefinite number of outfits using the same network, and even generalizing to unseen outfits. Afterwards, Chapter 5 will introduce a completely novel and interesting methodology for outfit animation. Inspired by Physically Based Simulation, I devise the first unsupervised deep-based methodology to train outfit-specific neural networks for cloth animation. By removing the need of data, this approach hugely reduces the time required to obtain animated outfit models. Moreover, the results qualitatively outperform all the related works in spite of using the simplest neural network architecture. It also achieves

never-seen-before execution times due to its simplicity. This solution became the first viable option for real-life applications thanks to its remarkable properties and establishes a promising research direction for the domain, neural cloth simulation. Finally, in Chapter 6, I dive deep into neural cloth simulation. On one hand, I offer for the first time a methodology able to learn cloth dynamics unsupervisedly. This complements the previous methodology, which is limited to quasi-static deformations. And on the other side, I provide of extensive analysis and insights in the specific domain of neural cloth simulation, which presents some particularities of its own. Next, I will summarize the main contributions of this thesis. Contributions are organized as separate self-contained chapters, each with its own sections for related works, methodology and results.

Chapter 2. SMPLR, a methodology for human 3D pose and shape retrieval. Human pose estimation, both in 2D and 3D, is performed by regressing the location of a set of body joints to form a *skeleton* (Eichner et al., 2012; Atrevi et al., 2017; Zhou et al., 2017; Luvizon et al., 2018). The concept of a simplified *skeleton* to describe pose comes from the computer graphics literature (Magnenat-thalmann et al., 1988). One can see how it would be desirable to extract not only 3D pose but also the body shape or geometry (Bogo et al., 2016; Kanazawa et al., 2018; Kolotouros et al., 2019). It is standard to use meshes, as a set of triangles, to represent 3D objects. This increases the complexity of the problem from predicting a few joints to thousands of vertices. For this reason, it is common to use parametric body models instead. For the chosen parametric model in this work, SMPL(Loper et al., 2015a), body pose is a 72-dimensional array and body shape a 10-dimensional array. This reduces the problem to a regression of only 82 parameters from an RGB image. While direct supervision over this parameters it is possible, since SMPL is differentiable, the most common solution is to append this model at the end of the neural network and backpropagate through it by supervising the 3D mesh directly. Nonetheless, this leads to unrealistic predictions with unrealistic pose and shape. Related works propose strong regularization terms to avoid bad predictions (Kanazawa et al., 2018). A more effective solution is to use intermediate representations (Omran et al., 2018; Pavlakos et al., 2018). This solution has been explored with 2D representations, which are suboptimal for inferring 3D, and using 3D joints only, which makes body shape and the orientation of some specific joints ambiguous. Here I propose a solution that uses as intermediate representation a set of 3D joints and landmarks placed across the body surface.

Chapter 3. CLOTH3D, a dataset of 3D draped humans and a baseline generative model trained on these data. After tackling the problem of human 3D pose and shape retrieval from images, the next necessary step is to move the scope towards garments. It is known that neural networks require huge volumes of data to be trained. Unfortunately, for the domain of clothing, such data is not publicly available at large scale. At the time, the only 3D garment datasets with realistic cloth behaviour were the BUFF dataset (Zhang et al., 2017), real data obtained with 3D scans, TailorNet data (Patel et al., 2020) and the dataset used in (Wang et al., 2018), both being synthetic data obtained through simulation. Nonetheless, these datasets are small in terms of garment variability, poses and body shapes while showing low to none cloth dynamics. Other related datasets are 3DPW (Marcard et al., 2018) and 3DPeople (Pumarola et al., 2019a), but these represent draped humans through rigid transformations only, and therefore, they lack of realistic cloth behaviour. Then, aiming to fill the gaps of the publicly available data, CLOTH3D is the first large scale dataset of its kind, containing over 2M samples, focusing in cloth dynamics and outfit and body shape variability. CLOTH3D is a syntehtic dataset generated

using Physically Based Simulation (PBS). This is preferred due to the constrained and expensive setup needed to gather real data, plus the additional required post-processing and intrinsic measurement error. Note that PBS has its drawbacks as well. It demands a significant amount of computational resources and it may present a slightly lower level of realism, specially for very complex cloth-to-body interactions. On the other hand, synthetic data has zero ground-truth error and has already been successfully used in deep learning (Nikolenko, 2019; Ros et al., 2016; Varol et al., 2017a). The baseline model proposed is inspired in related works (Alldieck et al., 2018a; Ma et al., 2019; Patel et al., 2020; Yang et al., 2018a). Garments are encoded uniformly on top of the human body as offsets from skin to cloth. Then, using a Graph Conditional Variational Auto-Encoder architecture, it is possible to learn a shared space for all garments and later generation conditioned on pose, shape, gender and *garment code*.

Chapter 4. DeePSD, a neural network capable of converting any 3D outfit mesh into an animation model. Some related works, including the baseline in the previous chapter, handle multiple garment types by encoding them as body offsets (Alldieck et al., 2018b; Alldieck et al., 2019; Bhatnagar et al., 2019; Patel et al., 2020). This has important limitations that manifest as poor quality predictions in which body geometry is transferred, surface is noisy and texturing shows artifacts due to over stretching or compression. Additionally, encoding cloth as body offsets bounds the solution to garments that take the form of a body homotopy and non-overlapping clothes. This mainly implies only that these methodologies can only be applied to a set (often just a few) of specific individual garments. In this chapter I re-formulate the problem. Instead of representing multiple garment types with a latent code, input parameters or using pre-defined classes to later animate them, I propose using as input the 3D mesh of the whole outfit instead. To do so, I design a novel fully-convolutional graph neural network that can take as input any 3D mesh, regardless of its vertex count, order or connectivity. As a first advantage, this removes the need of running an expensive lossy registration process of garments against the body. Using the original 3D outfit template also allows enforcing cloth priors to obtain a more realistic behaviour. Finally, I present a novel interesting property, generalization to unseen outfits.

Chapter 5. PBNS, the first methodology for unsupervised training of neural networks for outfit animation. It is common for neural networks to require big volumes of data to be trained. Data gathering is always expensive, and cloth domain is not an exception, whether it is with 3D scans or Physically Based Simulation. Unsupervised training for garment animation greatly reduces computational and economical cost. To achieve it, inspired by cloth simulation (Baraff et al., 1998; Liu et al., 2013; Macklin et al., 2016), I formulate the training as the optimization of an energy loss function based on physical laws that govern cloth behaviour. As it is common in the domain, outfits are skinned w.r.t. the underlying body and the network predicts deformations in rest pose. Back-propagating the energy loss creates some *forces* ($\mathbf{F} = -\nabla U$) that push the cloth vertices to the correct locations by updating the network weights. This is effectively a simulation, as garment deformations are *baked* into the network by physical forces. I call this Physically Based Neural Simulation (PBNS). Opposed to related works that propose complex network architectures to deal with the difficulties of learning realistic cloth deformations (Gundogdu et al., 2019a; Vidaurre et al., 2020; Patel et al., 2020), this approach works with the simplest form of neural network, a Multi-Layer Perceptron. The quality and robustness of the results allows to identify the flaws of supervised learning. Then, unsupervised garment animation is not only practical, but it is also a more suitable approach. Due

to its qualitative and computational performance, models trained with PBNS are the first viable option for real world applications. This approach is also the first to propose a formulation to explicitly handle cloth-to-cloth interactions, thus, the first to permit robust animation of multiple pieces and layers of cloth, instead of individual garments.

Chapter 6. Neural Cloth Simulation, as the first complete methodology to simulate garments into neural networks. Despite its novelty and success, PBNS is an incomplete solution. This is because it is limited to quasi-static deformations. For this reason, it is not a suitable solution for looser garments that often present rich dynamics. Motivated by this limitation, in this last chapter I describe how to bake garment dynamics into deep learning models. A first attempt to extend PBNS to the temporal dimension can be found in the work of (Santesteban et al., 2022), based on the optimization scheme proposed in (Liu et al., 2013; Martin et al., 2011; Gast et al., 2015). Unfortunately, their adaptation of the inertia loss term –which accounts for dynamics– to the deep learning framework incorrectly back-propagates forces back in time. In practice, their predictions show a lack of dynamics. On the other hand, I propose the first methodology for unsupervised learning of garment dynamics. Neural cloth simulation presents a set of peculiarities and specific challenges. In this chapter I also describe them along with in-depth analysis to establish the bases of the domain and support future research. Finally, I tackle the problem with a specific disentangled network architecture and input descriptors that allow learning different sub-spaces for static and dynamic garment deformations, which brings interesting properties: increased generalization, a novel motion augmentation for training and a never-seen-before motion control property during inference.

Chapter 7. In this final chapter I state the conclusions of the thesis as well as the obtained results, for each of the contributions and as a whole. The main goal to achieve in this thesis is, as the title describes it, an efficient and realistic animation of 3D garments through deep learning techniques. In the conclusions I will explain the progress made in each step of the way, how much has been achieved and how much is there to be achieved. This shall provide of a measure of my contribution in the domain. The footprint I left on it, which I hope is only the first of many. I also define what I believe the future of the research will be like for neural garment animation, which challenges need to be addressed and which could be promising solutions. Finally, we all have witnessed how deep learning has proven not only useful for many tasks but also growingly compelling, and I am eager to see what the future of the field will bring us.

As a result of this thesis I made the following publications in relevant venues:

- Meysam Madadi et al. (2020). "SMPLR: Deep learning based SMPL reverse for 3D human pose and shape recovery". In: *Pattern Recognition*, p. 107472
- Hugo Bertiche et al. (2020). "CLOTH3D: Clothed 3D Humans". In: *European Conference on Computer Vision*. Springer, pp. 344–359
- Meysam Madadi et al. (2021a). "Deep unsupervised 3D human body reconstruction from a sparse set of landmarks". In: *International Journal of Computer Vision* 129.8, pp. 2499–2512
- Meysam Madadi et al. (2021b). "Learning Cloth Dynamics: 3D+Texture Garment Reconstruction Benchmark". In: *Proceedings of the NeurIPS 2020 Competition and Demonstration Track*. Ed. by Hugo Jair Escalante et al. Vol. 133. Proceedings of Machine Learning Research. PMLR, pp. 57–76. URL: `https://proceedings.mlr.press/v133/madadi21a.html`

- Hugo Bertiche et al. (2021a). "Deep Parametric Surfaces for 3D Outfit Reconstruction from Single View Image". In: *2021 16th IEEE International Conference on Automatic Face and Gesture Recognition (FG 2021)*, pp. 1–8. DOI: 10.1109/FG52635.2021.9667017
- Hugo Bertiche et al. (2021b). "Neural Implicit Surfaces for Efficient and Accurate Collisions in Physically Based Simulations". In: *CoRR* abs/2110.01614. arXiv: 2110.01614. URL: https://arxiv.org/abs/2110.01614
- Hugo Bertiche et al. (2021d). "DeePSD: Automatic deep skinning and pose space deformation for 3D garment animation". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 5471–5480
- Hugo Bertiche et al. (2021c). "PBNS: Physically Based Neural Simulation for Unsupervised Garment Pose Space Deformation". In: *ACM Trans. Graph.* 40.6. ISSN: 0730-0301. DOI: 10.1145/3478513.3480479. URL: https://doi.org/10.1145/3478513.3480479

More information about the outcomes of my thesis in hbertiche.github.io. The more recent ones have a publicly available repository with the implementation of the methodologies. Code for Chapter 6 will be available there soon too.

Chapter 2

SMPLR: Deep learning based SMPL reverse for 3D human pose and shape recovery

2.1 Introduction

We have defined the goal of this thesis as efficient and realistic animation of 3D garments through deep learning. Nonetheless, as explained in Chapter 1, this deep learning journey begins in human 3D pose and shape recovery from RGB. 3D human pose estimation from still RGB images is a challenging task due to changes in lighting conditions, cluttered background, occlusions, inter and intra subject pose variability, as well as ill-posed depth ambiguity. Moreover, due to its nature, accurate annotation of captured data is not a trivial task and most available datasets are captured under controlled environments (Ionescu et al., 2014; Sigal et al., 2010; Mehta et al., 2017).

One case of human pose representation is 3D joint locations. However, 3D joints do not implicitly show the morphology of the body. Being able to estimate body shape or mesh along with joints allows a wide method applicability, including movie editing, body soft-biometrics measurements or cloth retexturing, among others. Besides, such a dense body representation may help to achieve more accurate estimations of 3D joints. Available body models range from simple geometrical objects, like compositions of cylinders and spheres, to complex parametric statistical models such as SCAPE (Anguelov et al., 2005) and SMPL (Loper et al., 2015a). SMPL (Bogo et al., 2016; Pavlakos et al., 2018; Kanazawa et al., 2018) generates realistic body meshes based on PCA shape components along with relative axis-angle rotations of joints. Rotations form body pose in a defined kinematic tree and are computed for joints with respect to their parent's nodes. The goal is to estimate SMPL parameters from an RGB image such that generated body mesh describes and fits as much as possible to the visible human in the image. This can be done by fitting generative models (Sigal et al., 2008; Guan et al., 2009; Lassner et al., 2017) or training discriminative deep models (Kanazawa et al., 2018; Omran et al., 2018; Varol et al., 2018). On the one hand, regular generative optimization solutions are shown to be sensitive to noise and need a careful and complex design of objective functions. On the other hand, while deep models have shown superior performance over the former solutions, they are data-hungry approaches. In both cases, a direct regression of SMPL parameters is a complex task because: 1) SMPL is a many-to-one complex function[1] which is sensitive to data noise (i.e. optimizing SMPL parameters may converge to

[1]SMPL details are given in the section 2.3.1

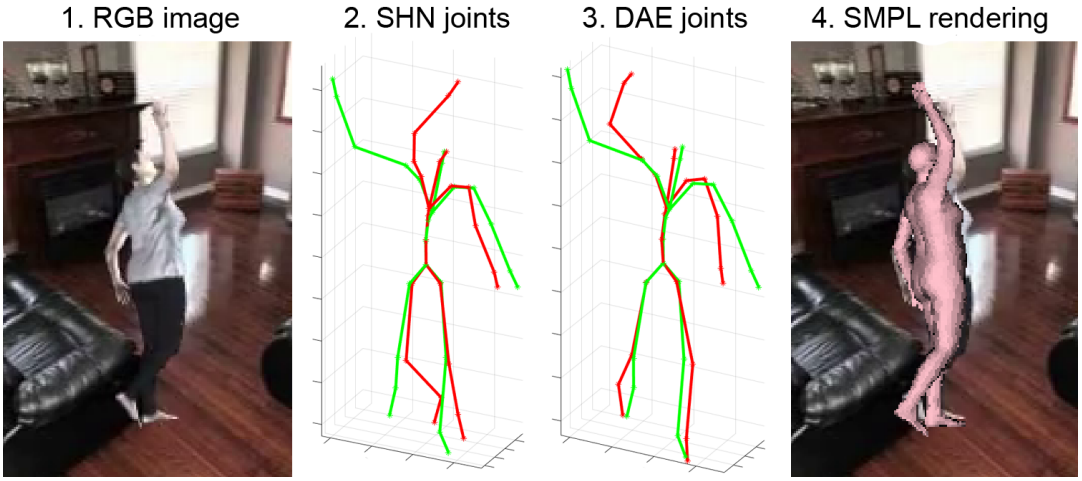


FIGURE 2.1: Illustration of the proposed model output. Given 1) an input RGB image, 2) an initial estimation of 3D joint locations is applied based on a given CNN (volumetric stacked hourglass (SHN) (Newell et al., 2016) in this case). Green line shows ground truth and red line shows 3D estimated joints. 3) Our denoising autoencoder model is able to recover pose from structured error. Finally, 4) body mesh is rendered by SMPL based on the proposed SMPL reverse strategy.

invalid values), and 2) accurate image annotation with SMPL pose and shape parameters in large in-the-wild datasets is infeasible. Therefore, researchers developed their solutions based on available 2D joints by applying intermediate representations (Pavlakos et al., 2018; Omran et al., 2018) or adversarial training (Kanazawa et al., 2018) for 3D inference. However, it is known that estimation of 3D data from 2D is ill-posed and can lead to sub-optimal solutions.

In this chapter, given an RGB image, we first estimate 3D joints and a sparse set of landmarks placed along body surface. Then, we use them to regress SMPL pose and shape parameters which are fed to SMPL model. The output is a detailed body mesh. We call this procedure SMPL reverse (SMPLR). One can imagine SMPLR as an autoencoder where latent embeddings are pose and shape components. We define encoder as a number of Multi-layer Perceptron (MLP) networks while decoder is SMPL. By first estimating 3D joints and landmarks, as an intermediate representation, 1) we avoid a direct regression of SMPL parameters, which easily yields non-realistic body meshes, 2) we can safely train SMPLR, even end-to-end, in a simple way without explicit constraints on SMPL, and 3) we provide flexibility to the design, i.e. SMPLR can be trained independently of RGB data using millions of generated mocap-like data, thus allowing cross-dataset generalization.

When 3D ground truth data is available for RGB images, any state-of-the-art CNN can be used to estimate 3D joints and landmarks. However, such ground truth data is not available for in-the-wild datasets. Besides, estimated 3D joints can have structured error due to depth ambiguity or occlusions. To handle these cases we design a denoising autoencoder (DAE) network (Vincent et al., 2010) as an extra module between CNN and SMPLR able to lift 2D joints to 3D and/or recover from structured error. We show the proposed model output in Fig. 2.1.

In summary, our main contributions are as follows:

- We build a denoising autoencoder that learns to recover input data from structured error. The model transforms 2D/3D joints to more human-consistent 3D joint predictions, enforcing symmetry and proportions on bone lengths.

- We design a two-branch MLP network to regress SMPL parameters from 3D joints and landmarks given by DAE. We refer to the combination of DAE, MLP and SMPL as SMPLR. This allows the inference of human body mesh from a sparse point representation. Finally, we gain an improvement over chosen CNN by end-to-end training with SMPLR.
- Throughout our experiments, we demonstrate that it is possible to obtain an accurate human body model from a set of joints and landmarks predictions. We obtain state-of-the-art results for SMPL-like architectures on Human3.6M (Ionescu et al., 2014) and SURREAL (Varol et al., 2017a) datasets.

2.2 Related work

In this section, we review state-of-the-art works on 3D human pose estimation from still RGB images.

Lifting 2D to 3D. Depth regression from 2D is an ill-posed problem where several 3D poses can be projected to the same 2D joints. (Atrevi et al., 2017) assign 3D joints from a dataset by 2D body silhouette matching, while (Chen et al., 2017) show that copying depth from 3D mocap data can provide a fair estimation when a nearest 2D matching is given. However, (Moreno-Noguer, 2017) shows that distance of random pairs of poses has more ambiguity in Cartesian space than Euclidean distance matrix. Recent works show that directly using simple (Martinez et al., 2017) or cascade (Hoang et al., 2018) MLP networks can be more accurate. Additionally, 2D joints can be wrongly estimated, making previous solutions sub-optimal. (Yang et al., 2018b) use adversarial training and benefit from available 3D data along with 2D data to infer depth information. In our case, the proposed denoising autoencoder is used to lift 2D pose to 3D in the lack of paired image and 3D ground truth data.

Direct regression refers to regressing 3D pose directly from an RGB image. Due to the nonlinear nature of the human pose, 3D pose regression without modeling correlation of joints is not a trivial task. (Brau et al., 2016) estimate 3D joints and camera parameters without direct supervision on them. Instead, they use several loss functions for projected 2D joints, bone sizes and independent Fisher priors. (Sun et al., 2017) propose a compositional loss function based on relative joints with respect to a defined kinematic tree. They separate 2D joints and depth estimation in the loss. We avoid relying on such complex losses by using 3D joints and landmarks as intermediate representation.

Probability maps are joints likelihood computed for each pixel/volume. From 2D joint heatmaps different solutions are applied to infer the third dimension. (Tome et al., 2017) iteratively minimize a function based on 2D belief map and a learnt 3D model to find the most likely 3D pose. Since probability maps are dense predictions, fully convolutional networks are usually applied. (Luo et al., 2018) extend stacked hourglass (SHN) (Newell et al., 2016) to estimate 2D joint heatmaps along with limb orientation map in each stack. (Pavlakos et al., 2017) extend SHN to output 3D volumetric data by a coarse-to-fine architecture where at each stack the third dimension is linearly increased with 2D heatmaps. (Nibali et al., 2018) propose marginal heatmaps from different axis viewpoints.

Pose and shape estimation. In order to compute a detailed body model, it is common to estimate body volume or mesh along with 3D pose. Early proposals were mainly based on the combination of simple geometric objects (Stoll et al., 2011; Sigal et al., 2012), while recent approaches are PCA-based parametric models like SMPL (Loper et al., 2015a). (Bogo et al., 2016) were the first to apply SMPL for body pose

and shape recovery. Their method was based on regular optimization procedures given an objective function with several constraints, minimizing projected joints and pre-estimated 2D joints. Such a complex function design is critical for the success of optimization procedures, since SMPL is a many-to-one function and sensitive to noise. (Lassner et al., 2017) extended the previous work by including a bi-directional distance between projected mesh and body silhouette. Recent works embed SMPL within deep models. (Tung et al., 2017) regressed SMPL pose and shape along with camera parameters. They trained the model with supervision on synthetic data and fine-tuned without supervision at inference time using 2D joints, silhouette and motion losses. Similarly, (Kanazawa et al., 2018) regressed as well SMPL and camera parameters, but in addition, they applied adversarial training. Predictions are fed to a discriminator network which classifies them as real/fake with respect to real body scans. Similar to our work, (Pavlakos et al., 2018; Omran et al., 2018; Zanfir et al., 2018) estimate pose and shape parameters from intermediate information, like body segments (Omran et al., 2018), 2D joint heatmaps and body mask (Pavlakos et al., 2018) or 3D joints (Zanfir et al., 2018). They include SMPL to obtain 3D joints and mesh which are used to compute the loss either in 2D (back-projected from 3D) or 3D. However, this process is ill-posed and sub-optimal because of the loss of depth information (in the case of (Pavlakos et al., 2018; Omran et al., 2018)) or the ambiguity in joint orientations and shape parameters (in the case of (Zanfir et al., 2018)). We show 3D joints and surface landmarks can better deal with this problem outperforming aforementioned solutions. Recently, (Varol et al., 2018) proposed a multi-tasking approach to estimate body 2D/3D pose, pixel segments and volumetric shape. They use SMPL to generate ground truth body volumes and do not embed SMPL function within the network.

In this chapter, we propose an approach to estimate 3D body pose and shape by the use of intermediate information and SMPL model. We can benefit from end-to-end training. Besides, our method can be adapted to 2D-to-3D solutions when just 2D ground truth data is available.

2.3 Methodology

We estimate 3D joints $\mathbf{J} = \{j\}_1^K$ and surface vertices $\mathbf{T} = \{t\}_1^C$ from a single RGB image I, where $j, t \in \mathbb{R}^3$, and K and C are the number of body joints and surface points, respectively. We define $\mathbf{L} \subset \mathbf{T}$ as a set of sparse surface landmarks and $\mathbf{J}^+$ as a concatenation of the two matrices. In order to compute a detailed mesh, we use SMPL model (Loper et al., 2015a). Our goal is to estimate SMPL parameters from image I using deep learning without directly regressing them. This way, we avoid possible artifacts while keeping the architecture flexible in the lack or presence of 3D ground truth data. Our network contains three main modules shown in Fig. 2.2. First, joints and landmarks locations are estimated by any chosen CNN ($\mathbf{J}^+{}_{CNN}$). Afterwards, DAE filters structured error or lifts 2D to 3D ($\mathbf{J}^+{}_{DAE}$). Finally, SMPLR recovers pose and shape from the predictions of the previous module and reconstructs a detailed body mesh and skeleton. Next, we explain DAE and SMPLR in detail.

2.3.1 SMPL review

SMPL is a statistical parametric function $M(\beta, \theta; \Phi)$ which maps shape parameters β and axis-angle pose parameters θ into vertices $\mathbf{T}$, given learnt model parameters Φ. Given a template average body mesh with vertices $\mathbf{T}^*$ and a dataset of

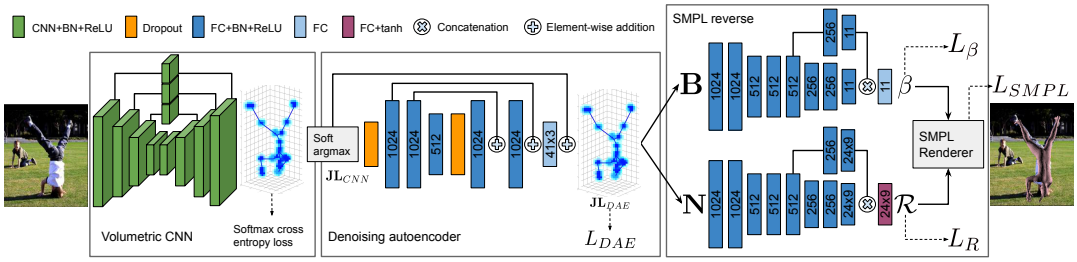


FIGURE 2.2: System pipeline. A CNN estimates volumetric heatmaps. Soft argmax converts heatmaps to joints locations and feeds them to denoising autoencoder module. Soft argmax is differentiable, thus, gradients can be backpropagated. Finally, we compute normalized relative distances **B** (eq. 2.3) and normalized relative joints **N** (eq. 2.1) which are fed to two independent networks designed to regress SMPL parameters. At the end SMPL is responsible to render a realistic body mesh. Dashed arrows show where the loss is applied.

scanned bodies, two sets of principal components $\mathcal{S} = [\mathbf{S}_1, ..., \mathbf{S}_{|\beta|}] \in \mathbb{R}^{3C \times |\beta|}$ and $\mathcal{P} = [\mathbf{P}_1, ..., \mathbf{P}_{9K}] \in \mathbb{R}^{3C \times 9K}$ are learnt to form model parameters Φ (where $|\beta| = 10$, $C = 6890$ and $K = 24$). Then, template shape vertices $\mathbf{T}^*$ can be morphed to $\mathbf{T}_s^*$ by $vec_{3,C}^{-1}(\mathcal{S} \times \beta) + \mathbf{T}^*$ where $vec^{-1}(.)$ is a reshaping operator. Template 3D joints must be updated as well w.r.t. the body shape. This is done by a regressor matrix $\mathcal{J}$ (as part of parameters Φ) from updated vertices $\mathbf{T}_s^*$. Bases $\mathbf{P}_i$ are responsible for small pose-based displacements due to body soft-tissue behavior and have small contribution in the shape deformation. Given a kinematic tree (i.e. Fig. 2.3) a set of relative rotation matrices $\mathcal{R} = [\mathbf{R}_1, ..., \mathbf{R}_K] \in \mathbb{R}^{3 \times 3}$ are computed for each joint with respect to their parents. Each $\mathbf{R}_i$ is a function of $\theta_i \in \mathbb{R}^3$ and is computed based on Rodrigues formulation. These rotation matrices are mainly used because of two reasons: i) to pose the mesh by rotating body parts relatively in the kinematic tree, and ii) to update the template shape in rest pose θ^* by basis $\mathbf{P}_i$. Please read (Loper et al., 2015a) for more detailed explanations of the SMPL.

SMPL model has several characteristics. First, it is differentiable, which yields the possibility to be used along with deep networks. Secondly, it does not constrain invalid pose and shape values and, thus, it is a many-to-one function. This means that given an RGB image, end-to-end training of a CNN from scratch with SMPL attached on top may converge to a non-optimal solution. One of the main reasons of this is the usage of Rodrigues formulation and axis-angle pose parameters, as it is known not to be unique (periodicity of θ). A possible solution is to directly use rotation matrices as proposed in (Lassner et al., 2017; Omran et al., 2018). Finally, SMPL is a generative model which allows us to generate mocap-like data for free. Also, it can be used to generate synthetic realistic images (Varol et al., 2017a).

2.3.2 SMPL reverse

A natural way of embedding SMPL in a deep network is to estimate β and θ given image I and feed them to SMPL. However, this is a challenging task because of the aforementioned many-to-one property, plus the noise sensitivity of the model. Besides, direct regression of SMPL parameters may generate artifacts (Kanazawa et al., 2018; Tan et al., 2017). Instead, researchers use intermediate representations like 2D joints and body silhouette (Pavlakos et al., 2018) or body segments (Omran et al., 2018) to regress SMPL parameters. Although such data is easier to annotate from RGB images than SMPL data, they provide sub-optimal mapping to SMPL parameters, because 1) estimating 3D from 2D is an ill-posed problem, and 2) the loss is

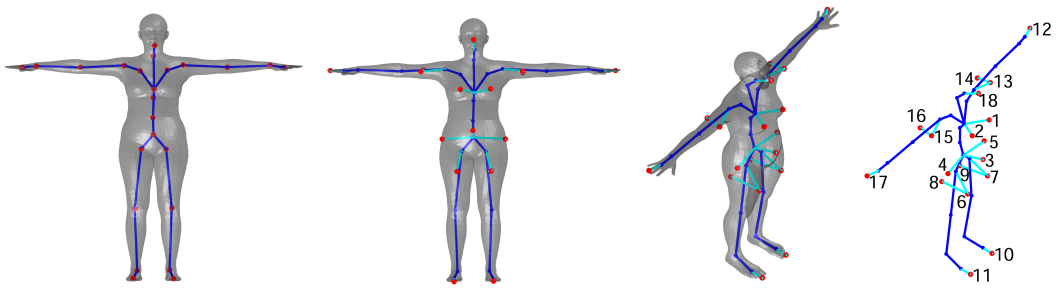


FIGURE 2.3: SMPL kinematic tree for a) joints, b) and c) proposed landmarks. d) Assigned numbers to the landmarks. Pelvis is set as root. Points are shown in red.

computed from noisy back-projected 2D estimations. In this chapter, we instead propose an autoencoder-like scheme, i.e. the input 3D data is recovered in the output, while pose and shape are obtained in the encoder and SMPL is taken as decoder. We refer to this model as SMPL reverse (SMPLR, see Fig. 2.2).

This design has several benefits: SMPLR can be trained 1) without the need of constraints on SMPL, 2) independent to RGB data using millions of generated 3D mocap-like data, and 3) end-to-end with a CNN. All of these provide simplicity and flexibility in the design and training of the entire network. In the results section we show that SMPLR formulation can generate more accurate estimations than state-of-the-art SMPL-based alternatives. Furthermore, SMPLR acts as a strong regularization on CNN model when trained end-to-end and it enhances the internal coherence among joints for CNN predictions. In sec. 2.4.5 we propose an effective incremental training for this task.

We model SMPLR encoder with deep MLP networks. We design two independent networks $\mathcal{R} = \Omega(\mathbf{N}; \phi_p)$ and $\beta = \Psi(\mathbf{B}; \phi_s)$ with the same architecture (see Fig. 2.2 for details) for pose and shape estimation, respectively, where $\phi_.$ corresponds to network parameters, $\mathbf{N}$ is a vector of normalized relative joints and $\mathbf{B}$ is a vector of normalized relative distances. A reason for the choice of two networks is that $\mathcal{R}$ and β are independent variables. Besides, we want the encoder to be cross-dataset applicable and explainable w.r.t. the pose and shape parameters. In available datasets, while pose parameters have a high variability, there is no much variability in shape parameters. For instance, Human3.6M dataset (Ionescu et al., 2014) only consists of 11 subjects and training Ψ is not feasible by relying just on this dataset. In the results section we show the contribution of each network to the final joint estimates.

Since we define SMPLR as an autoencoder, its input must be $\mathbf{J}$ and $\mathbf{T}$. However, $\mathbf{T}$ is a high dimensional vector and all vertices do not necessarily contribute equally in the computation of pose and shape parameters, wasting network capacity if all of them are considered. To cope with this issue, we empirically select a subset of points as landmarks $\mathbf{L} \subset \mathbf{T}$, which represent body shape and complement $\mathbf{J}$. Without landmarks, network converges to the average body fatness and the problem still remains ill-posed due to the ambiguity of joints orientation. Landmarks help to cope with these two problems. Besides, it is cheaper to gather landmarks in mocap datasets rather than scanning the whole body. We show the 18 selected landmarks and their assigned kinematic tree in Fig. 2.3. Next, we explain networks details.

$\mathbf{J}$ and $\mathbf{L}$ are concatenated to form $\mathbf{J}^+$ which has been estimated beforehand by DAE (i.e. $\mathbf{J}^+{}_{DAE}$). For the easiness of the reading we omit the subscript $._{DAE}$ from

$\mathbf{J}^+{}_{DAE}$ in the next formulations. Given a kinematic tree $\kappa \in \mathbb{R}^{42}$, we define $\mathbf{N}$ as:

$$\mathbf{N}_i = \frac{\mathbf{J}^+{}_i - \mathbf{J}^+{}_{\kappa(i)}}{\|\mathbf{J}^+{}_i - \mathbf{J}^+{}_{\kappa(i)}\|_2}, \text{ for } i \in [2..42], \tag{2.1}$$

where $\kappa(i)$ defines parenthood indices. The reason for this normalization is that, in order to compute relative joint rotation $\mathbf{R}$, we do not need to know relative distances. This frees network capacity from unnecessary data variability. Such relative distances are embedded in the computation of shape parameters. Thus, given relative distances $\mathbf{B}^*$ computed from template joints and landmarks $\mathbf{J}^{+*}$, we define $\mathbf{B}$ as:

$$\mathbf{B}^*_i = \|\mathbf{J}^{+*}_i - \mathbf{J}^{+*}_{\kappa(i)}\|_2, \text{ for } i \in [2..42], \tag{2.2}$$

$$\mathbf{B}_i = \|\mathbf{J}^+{}_i - \mathbf{J}^+{}_{\kappa(i)}\|_2 - \mathbf{B}^*_i, \text{ for } i \in [2..42]. \tag{2.3}$$

SMPL originally provides two different models for male and female. Selecting a proper gender model for each sample has a crucial impact on the accuracy at inference time. Therefore, we also include gender as an extra term on the shape parameters β. We assign gender a value from the set $\{-1, +1\}$ and learn it as a regression problem along with other shape parameters.

Loss function. $\mathcal{L}_2$ loss has been commonly applied in recent regression problems (Tan et al., 2017; Pavlakos et al., 2018). However, we found $\mathcal{L}_2$ loss to have problems in convergence and generalization in case of noisy inputs. Instead we use $\mathcal{L}_1$ loss on $\mathcal{R}$ and β, called L_R and L_β, to supervise Ω and Ψ networks. Firstly, this is done isolated from SMPL, which means no back-propagation is applied through SMPL. This is important for a stable training and fast convergence. Then, for performance gains, we fine-tune the networks adding $\mathcal{L}_1$ loss on SMPL output, called L_{SMPL}. SMPL output contains $\mathbf{J}$ and $\mathbf{T}$. However, to have SMPLR architecture resembling an autoencoder, we compute L_{SMPL} on landmarks rather than the whole $\mathbf{T}$. The final SMPLR loss is $L_R + L_\beta + L_{SMPL}$.

2.3.3 Denoising autoencoder

Estimated joints by any CNN may have structured noise. For instance, in the case of occluded joints the error is higher due to their ambiguity and the lack of visual evidence. Visible joint predictions have as well structured error, following a Gaussian distribution. Such structured or Gaussian noise can be detected and learnt explicitly, helping to further improve initial estimation of $\mathbf{J}^+{}_{CNN}$ to be fed into SMPLR module. Denoising autoencoder networks (Vincent et al., 2010) are useful tools for such scenario, being able to learn structured patterns of the input better than ordinal autoencoders.

In this chapter, we propose a DAE network as a bridge between CNN backbone and SMPLR. With the proposed DAE we are able to denoise 3D joints and landmarks. This procedure can be critical for error-prone CNNs, such as shallow networks. Moreover, it can be detached from CNN and trained independently given a large amount of mocap or synthetic SMPL-generated data. However, DAE may not generalize well to the noisy test data if it is trained with noise-free ground truth data. Therefore, it is important to train DAE with adversarial noise in this scenario for generalization purposes. In the section 2.4.4 we show it is possible to train DAE with constrained uniform or Gaussian noise mimicking structured error without loss of generalization. It is also possible that only 2D joints are annotated in a given dataset. In sections 2.4.4 and 2.4.6 we also show DAE can lift 2D estimations to 3D.

This could also be done by the use of mocap or synthetic 3D data projected to 2D and adversarial noise. The architecture of DAE is shown in Fig. 2.2. We apply two dropouts, one right after the input and the other after last encoder layer. By applying skip connections between encoder and decoder layers we force the network to learn noise structure in a fast and stable way. The input to DAE is the initial estimation of ${\mathbf{J}^+}_{CNN}$ and the output is denoised ${\mathbf{J}^+}_{DAE}$. We apply $\mathcal{L}_1$ loss (so called L_{DAE}) on ${\mathbf{J}^+}_{DAE}$ to train this network. In order to force the awareness of adjacent joints correlations, we applied an $\mathcal{L}_1$ loss on the relative joints ($\mathbf{B}$ in Eq. 2.3) as well. However, we observed no significant impact on the results.

2.4 Experiments

This section describes training details, datasets, and evaluation protocol. Then, we perform an ablation study of the different model components and compare it against state-of-the-art alternatives.

2.4.1 Training details

We build our backbone CNN based on the well-known stacked hourglass network (SHN) (Newell et al., 2016) using 5 stacks. We extend final layers of each stack to volumetric heatmaps, i.e. including an extra dimension to discretize depth of the joints into 16 bins. The output of each stack is a tensor of size $64 \times 64 \times 16 \times 41$, where 64 is the size of X-Y axis and $41(= 23 + 18)$ is the number of joints and landmarks. We train this network with softmax cross entropy loss. All models and experiments were implemented on TensorFlow and trained on a GTX 1080 Ti. We used Adam optimizer in all the experiments with a learning rate of 0.01 for SHN and 0.001 for DAE, Ω and Ψ networks. All networks are trained from scratch using a Xavier initializer. SHN converged in 150-250 epochs with batch size 6-10 samples. The rest of networks in the ablation analysis were trained with batch size 256. We used a keeping probability 0.8 for dropout layer in DAE.

Preprocessing. Images are cropped to a square. To do so, we assume camera focal length and object distance to camera is available beforehand. First, the corners of a 2.5×2.5m grid, centered at average joint location and perpendicular to camera axis, are projected to image plane and define the cropping area. Then, cropped images are scaled to network input size (256×256). This enforces a proportionality among pixel and real world sizes, larger people will appear bigger in image space as well. In those cases where the crops land outside the frame, a random image from VOC Pascal dataset is used for padding. Following (Varol et al., 2018) we use ground truth focal length and object distance to the camera in all experiments, both in training and inference time. However, to study the impact of scale ambiguity on the 3D joint prediction, we also estimate the cropping area in Human3.6M (Ionescu et al., 2014) dataset and show the results (see section 2.4.6).

End-to-end training. We applied incremental training. First, all networks (i.e. SHN, DAE, Ω and Ψ) were trained independently and then the whole network was fine-tuned end-to-end. In the ablation study we analyze the effect of different combinations of modules in the training.

2.4.2 Datasets

UP-3D (Lassner et al., 2017). This dataset was designed by fitting a gender neutral SMPL model into images from LSP, LSP-extended and MPII-HumanPose datasets,

	Model	Hd	Ts	Sr	Ew	Wt	Hp	Kn	Ft	Avg. Jt	Avg. Lm	Avg. Bn
CNN	$Alexnet$	100.0	41.6	99.0	179.9	246.9	34.6	138.5	217.3	133.0	-	31.9
	SHN_{nL}	47.5	23.0	44.2	77.2	112.0	16.3	61.9	102.7	62.8	-	10.4
	SHN	46.2	23.0	43.0	75.5	110.2	15.3	61.2	102.2	59.9	61.5	9.3
	SHN_{e2e}	45.1	22.2	43.3	74.4	108.2	16.0	57.9	94.2	57.8	59.6	**9.0**
	SHN_{aug}	45.0	22.3	41.1	72.7	105.7	14.4	59.6	99.8	57.5	59.3	9.0
	SHN^{final}	**40.8**	**20.9**	**38.0**	**66.8**	**93.4**	**14.3**	**55.7**	**92.9**	**53.0**	**54.3**	9.7
DAE	$DAE_{Alexnet}$	89.3	37.1	87.2	160.8	230.8	29.6	131.7	205.9	121.5	-	22.7
	DAE_{SHN}	**45.8**	**22.2**	**42.2**	**75.4**	**108.5**	**14.4**	**61.8**	**103.2**	**59.2**	**61.1**	**9.5**
	DAE^{2d}_{SHN}	51.4	23.1	46.2	83.7	121.7	15.1	66.6	115.4	65.2	66.1	11.8
SMPLR	Ψ	16.5	9.1	13.8	17.3	19.9	5.8	11.7	21.3	14.4	11.5	6.6
	Ω	55.1	22.3	48.8	85.8	127.1	13.4	68.6	122.3	67.8	-	-
	Ω_{smpl}	**50.1**	**20.1**	**44.9**	**83.6**	**123.6**	**12.4**	**63.9**	**111.9**	**63.8**	-	-
	single channel	58.7	24.3	53.6	94.1	142.5	16.0	68.8	116.7	71.8	74.9	8.6
ALL	ALL	57.9	**24.0**	52.8	92.7	140.4	**15.8**	67.8	115.0	70.8	73.8	8.4
	ALL_{Proc}	**53.7**	26.1	**49.7**	**86.2**	**129.9**	21.6	**67.0**	**109.2**	**67.8**	**70.6**	**7.7**

TABLE 2.1: Ablation study of model components. Error in mm. Hd:Head, Ts:Torso, Sr:Shoulder, Ew:Elbow, Wt:Wrist, Hp:Hip, Kn:Knee, Ft:Foot, Jt:Joints, Lm:Landmarks and Bn:Bone length, $\{.\}_{nL}$: training without landmarks, $\{.\}_{aug}$: training with data augmentation, $\{.\}^{final}$: training with limb heatmaps and data augmentation, $\{.\}_{Alexnet}$: $Alexnet$ estimations as input, $\{.\}_{SHN}$: SHN estimations as input, $\{.\}^{2d}$: input depth is set to 0, $\{.\}_{smpl}$: training with $L_R + L_{SMPL}$ loss, $\{.\}_{e2e}$: model after end-to-end training, $\{.\}_{Proc}$: results after Procrustes mapping. Best results are bolded.

keeping samples with better estimates. This yields a total of 8515 labeled images in the wild, splitted into 5703 for training, 1423 for validation and 1389 for test. Every sample is provided with 2D joints annotations and SMPL parameters.

SURREAL (Varol et al., 2017a). Synthetic dataset of humans generated with SMPL model, containing exact annotations. It is composed of 68K videos containing SMPL generated humans moving on top of random backgrounds. For sampling, we skip a frame if the average joint distance is lower than 5cm w.r.t. to last sampled frame. This results in 2.8M training, 27K validation and 665K test samples.

Human3.6M (Ionescu et al., 2014). Human3.6M is a large dataset offering high precision 3D data thanks to MoCap sensors and calibrated cameras. It is composed of RGB videos of 11 subjects performing 15 actions twice while being recorded from 4 different viewpoints. It contains around 3.6 million frames. We sampled 1 of every 5 frames, ending with 312K training and 110K validation samples. Following state-of-the-art works, we use subjects S1, S5, S6, S7 and S8 for training, and S9 and S11 for testing. We generated ground truth SMPL parameters from the 3D data and body scans available in the dataset. Body scans allow an accurate estimation of shape parameters, computed only once per subject (shape does not change in short periods of time). Afterwards, we empirically defined a correspondence between SMPL joints and available 3D MoCap data. This matching is not perfect for some joints, which are weighted between $[0.25, 0.75]$ empirically to provide good estimations. This weighted matching allows optimization of pose through an $\mathcal{L}_2$ loss. Finally, correspondence of back joints is not accurate, so instead, we use a loss that penalizes unrealistic back bends, by correcting back's pose parameters that land outside an empirically defined symmetrical range centered at 0.

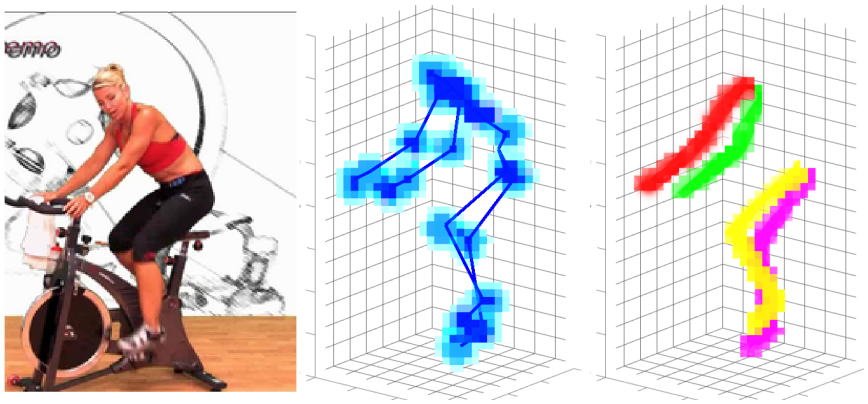


FIGURE 2.4: Sample volumetric heatmap of joints (middle) and limbs (right), each limb coded with a different color.

2.4.3 Evaluation protocol

We evaluate the models by mean per joint position error (MPJPE) in millimeters (mm). The same metric is extended to surface points to report error of the generated body meshes. Following related works we apply two protocols: **Protocol 1** where all joints/points are subtracted from the root joint and, **Protocol 2** where estimated joints are aligned with ground truth through Procrustes analysis. We also report mean Intersection over Union (IoU) on body silhouette after mesh projection to the image plane.

2.4.4 Ablation study

In this section, we study different components of the proposed model on SURREAL validation set. For this task we subsample the training dataset into 89K frames such that every pair of samples has at least one joint displaced 150mm w.r.t. to each other, thus enforcing a uniform distribution over the whole dataset. We use the setup in Sec. 2.4.1 to train each component. We explored several combination strategies during training to see the impact of each on the validation set. Except end-to-end training, all building blocks are trained isolated from the rest. We show results and description of each module in Tab. 2.1.

CNN backbone

We first evaluate the performance of the CNN backbones. The results are shown in Tab. 2.1 under **CNN** row. We first train a baseline *Alexnet* to regress 3D joints (without landmarks) using L_2 loss, Adam optimizer, learning rate 0.01 and batch size 32. We chose *Alexnet* for two reasons: 1) to compare the results with the proposed volumetric SHN, and 2) it is a shallow network and prone to have structured error so that we can study DAE impact as well. As expected *Alexnet* is not performing well to directly regress 3D joints. We then train volumetric SHN to predict $\mathbf{J}^+$ and $\mathbf{J}$ (so-called SHN and SHN_{nL}, respectively). As a result, landmarks help SHN to gain 3mm improvement over SHN_{nL}. Next we explain our contributions to the default volumetric SHN.

Final volumetric heatmap model. To evaluate our method against state-of-the-art we extend default volumetric SHN to include limb heatmaps in the output and train the model using data augmentation. Besides regular data augmentation (including random color noise, flipping and rotation), two extra methods are applied:

random background and artificial occlusion. By using binary masks for subjects provided at each frame, we remove the default background and replace it with a random image from VOC Pascal dataset. Similarly, we place random objects from VOC Pascal on random locations of the image to artificially create occlusions (Sańdi et al., 2018). In both cases we do not use images containing humans. SHN trained with data augmentation is called SHN_{aug} which shows 2.4mm improvement over SHN for average joint error (see Tab. 2.1).

Limb heatmaps are 4 additional volumetric heatmaps in the outputs of SHN. These heatmaps correspond to limb representations, created by composing segments from joint to joint (see Fig. 2.4). By fitting these heatmaps we expect to enforce the model to learn spatial relationships among joints to improve generalization. We train SHN_{aug} with limbs heatmaps and call this model SHN^{final}. The results displayed in the Tab. 2.1 show how these simple limb heatmaps indeed enhance the performance of SHN_{aug} by about 4.5mm on average joint error.

Denoising autoencoder

We also evaluate DAE trained with different inputs in several scenarios. Results are shown in Tab. 2.1 under **DAE** row.

Could we train DAE independent to SHN? Since DAE sequentially appears after SHN, it receives estimations from SHN. To answer this question we train DAE, as input, with i) 3D ground truth joints plus uniform noise with adapted bounds for each joint and ii) 3D joints estimated by SHN (so-called DAE_{noise} and DAE_{SHN}). We then evaluate both models with SHN estimations as input at test time. As a result, DAE_{noise} has an average error of 61.7mm (not shown in the table) which is similar to DAE_{SHN} (61.9mm). This shows the generalization ability of DAE.

Is DAE able to recover from structured noise? Other than DAE_{SHN}, we also train and test DAE with *Alexnet* estimations (called $DAE_{Alexnet}$). For *Alexnet* predicitons, DAE improves the error by 11mm, while on SHN the improvement is 0.7mm. This shows the ability of DAE to learn structured error.

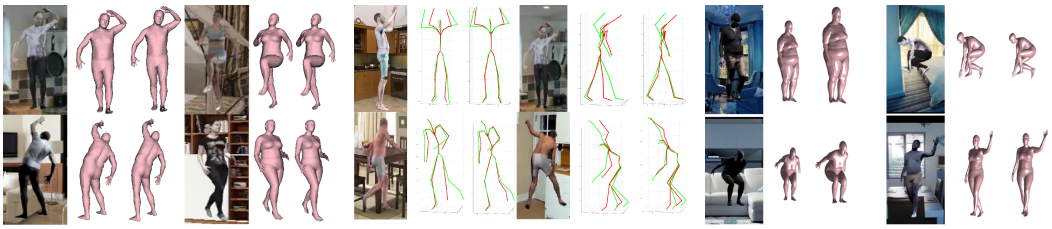
Is DAE able to lift 2D joints to 3D? To answer this question, we train and test DAE, following (Martinez et al., 2017), with SHN estimations while depth is set to 0 (called DAE^{2d}_{SHN}). In fact, we want to test how DAE performs in the lack of 3D ground truth data. As a result, the average error is slightly higher than 65mm. Although the average error is 3mm higher than SHN, it shows DAE can lift 2D pose to 3D with successful results. We note that training DAE^{2d}_{SHN} converges way slower than DAE_{SHN}.

SMPLR

In this section, we evaluate different components of SMPLR using SHN estimations as input in both training and test. The results are shown in Tab. 2.1 under **SMPLR** row. We first evaluate the impact of shape and pose estimations isolated from each other within SMPLR. In test time, shape and pose estimations are fed into SMPL to evaluate final joints error.

Shape estimation. We train Ψ network with L_β loss. During test we feed estimated β along with ground truth $\mathcal{R}$ to SMPL. The results are shown as Ψ in Tab. 2.1. As one can see shape estimation has a low impact on final error (around 14mm avg. joints error).

Pose estimation. We train Ω network first with L_R loss and then fine-tune it with $L_R + L_{SMPL}$ loss. The results are shown as Ω and Ω_{smpl} in Tab. 2.1, respectively.



(A) (B) (C)

FIGURE 2.5: Qualitative results of the ablation analysis. a) Visualization of the improvement on shape estimation due to landmarks. Left: image. Middle: estimation without landmarks. Right: estimation with landmarks. b) Prediction improvement due to end-to-end training. Left: image. Middle: prediction before end-to-end training. Right: prediction after end-to-end training. Green and red skeletons correspond to ground truth and predictions, respectively. c) Random samples with mistaken gender or viewpoint. Left: image. Middle: SMPL mesh. Right: SMPL mesh after Procrustes.

During test we feed estimated $\mathcal{R}$ along with ground truth β to SMPL. As a result we gain 4mm improvement in pose estimation by applying $L_R + L_{SMPL}$ loss. In general, the higher source of errors in SMPLR is in pose parameters rather than shape.

Impact of landmarks. Landmarks provide more visual evidence to the CNN when they are available in the dataset. Comparing SHN to SHN_{nL} in Tab. 2.1, one can see landmarks improve head, arms and hip estimations. We also train Ψ network with and without landmarks. Some qualitative results are shown in Fig. 2.5a.

Gender. We evaluate the accuracy of gender estimation in Ψ and achieve 89.5% accuracy. Such a high accuracy is critical for SMPL rendering. This means a given vector of shape parameters is interpreted differently by each gender model, i.e. a correctly estimated shape parameter but wrong gender estimation produces a wrong mesh generation, introducing a high error in SMPL mesh.

SMPLR architecture analysis. We compare two stream network with a single channel one. We keep the same architecture as Ω for the single channel network outputting shape and pose parameters in a single vector with size $11 + 24 \times 3 \times 3$. We directly feed estimated joints and landmarks to the network and train it similar to Ω_{smpl}. As a result, single channel network has around 1mm higher error than the proposed two stream network (see Tab. 2.1). We also study the impact of the proposed skip connection (inspired by inception module A). The SMPLR without skip connections performs 5mm worse than the proposed network for average joint error.

2.4.5 End-to-end training

Here, we describe how the end-to-end training was performed. Thanks to soft-argmax, the model is differentiable and trainable end-to-end. We first explore a regular end-to-end training by stacking already trained models SHN, DAE_{SHN}, Ψ and Ω_{smpl} along with SMPL on top. The loss is a summation of all intermediate losses. The order of magnitude of SHN loss is several times lower than the other losses. Therefore, without a proper balancing, the weights of the SHN vanish after few training steps. We empirically set this balance to be around $1e$-5. Fine-tuning is performed with a low learning rate, empirically set to $1e$-4, to ensure learning stability. We observed this model does not show improvements. Therefore, we propose the next procedure for end-to-end training.

	Prot. 1	Prot. 2
Bogo et al., 2016	-	82.3
Lassner et al., 2017	-	80.7
Tung et al., 2017*	98.4	-
Pavlakos et al., 2018	-	75.9
Omran et al., 2018	-	59.9
Kanazawa et al., 2018	87.9	56.8
Kolotouros et al., 2019	74.7	**51.9**
ALL	67.9	52.2
ALL_{Proc}	**62.6**	52.2

(A)

	Prot. 1	Prot. 2
Tome et al., 2017	88.4	70.7
Pavlakos et al., 2017	71.9	51.9
Zhou et al., 2017	64.9	-
Martinez et al., 2017	62.9	47.7
Sun et al., 2017	59.1	48.3
Sun et al., 2018**	64.1	-
Fang et al., 2018	60.4	**45.7**
SHN^{final}	61.3	50.1
SHN^{final}_{e2e}	**56.5**	46.3

(B)

TABLE 2.2: State-of-the-art MPJPE error in mm. on Human3.6M for both protocols. Best results are bolded. (a) SMPL-based methods comparison. SMPLR outperforms all SMPL-based methods. SMPL surface errors on this dataset are 88.2 and 81.3mm for ALL and ALL_{Proc}, respectively. (b) 3D pose comparison. The simple proposed updates on SHN show state-of-the-art-results after end-to-end training without using any extra data. * (Tung et al., 2017) use 32 joints. ** (Sun et al., 2018) report the results with and without extra data in the training. For a fair comparison we take this number from where no extra data has been used.

	SMPL surface	3D joints
Tung et al., 2017	74.5	64.4
Varol et al., 2018 (independent)	74.5	46.1*
Varol et al., 2018 (multi-task)	65.8	**40.8***
SHN^{final}	-	42.8*
SHN^{final}_{e2e}	-	**40.8***
ALL	66.0	50.6
ALL_{Proc}	**62.3**	48.2

TABLE 2.3: Errors (mm) on SURREAL dataset (protocol 1). Best results are bolded. .* are intermediate estimated 3D joints used to predict SMPL surface.

	SURREAL	Human3.6M	UP-3D
Varol et al., 2018 (multi-task)	-	-	0.73
ALL_{Proc}	**0.75**	**0.71**	**0.77**

TABLE 2.4: Silhouette IoU on three datasets.

We first train DAE, Ψ and Ω_{smpl} with ground truth 3D joints until they overfit on training data. Once trained, they are frozen and appended to SHN. Fine-tuning is performed with the same loss balancing and learning rate as before. The network shows improvement after the first training epoch, and after an additional epoch it fully converges. To ensure that this improvement is the result of the proposed end-to-end training, we trained SHN alone for more than 10 additional epochs without showing any improvement. The results of SHN after end-to-end training in Tab. 2.1 (called SHN_{e2e}) shows more than 2mm improvement. Some qualitative results are shown in Fig. 2.5b.

Recover SMPLR error. We stack all trained models to report final SMPL predictions (row **ALL** in Tab. 2.1). SMPLR naturally has a generalization error. Wrongly

FIGURE 2.6: Qualitative results. Top: SURREAL. Bottom: Human3.6M. Last sample of each row shows a failure case due to inaccurate pose estimation produced mainly by occlusion and/or background confusion.

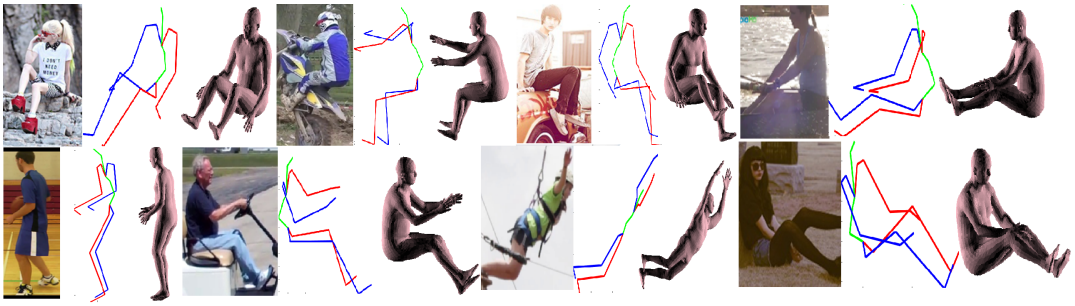


FIGURE 2.7: UP-3D results. Based on estimated 2D joints (middle) we compute 3D joints and render mesh (right).

estimated gender is a source of error in SMPLR. Not only gender, but also global rotation error embedded in Ω can degrade the results. Fortunately, the mesh can be partially corrected by an affine transformation as a post-processing. To do so we apply Procrustes mapping from SMPLR output to its input $\mathbf{J}^+$ and update the mesh accordingly. The results in Tab. 2.1 shows 4mm error recovery of ALL_{Proc} vs. ALL. Some qualitative examples are shown in Fig. 2.5c. Note that for this post-processing step, no ground truth information is required, as we align SMPLR output with SHN output (both model predictions), knowing that SHN is more accurate. This is different from protocol 2, where final predictions are aligned with ground truth.

2.4.6 State-of-the-art comparison

Human3.6M. We compare our results to state-of-the-art on Human3.6M in Tab. 2.2, split in two sets: SMPL-based solutions (2.2a) vs. only 3D pose recovery (2.2b). In the former, our proposed ALL_{Proc} outperforms state-of-the-art, especially in protocol 1 improving (Kolotouros et al., 2019) over 12mm. We note that we use Ψ network trained on SURREAL dataset to estimate shape parameters, since Human3.6M contains just 5 subjects in the training set (therefore, only 5 shapes). The results in the second set show that our simple modifications to SHN achieve state-of-the-art results after end-to-end training (SHN_{e2e}^{final}). Compared to (Pavlakos et al., 2017), a fixed small depth resolution of 16 bins in the volumetric heatmap works better than a coarse-to-fine setup. As we mentioned earlier, we also want to study the results when image cropping is not based on ground truth data. Therefore, we estimate camera focal length and object distance to camera following (Sařańdi et al., 2018) and use them to compute the cropping. The error on cropped image is less than 5px on each corner. To be robust against this error we fine-tune SHN^{final} with an additional random scaling image augmentation. As a result, the 3D joints error is

	Batch size	2D SHN	Vol. SHN	DAE	SMPLR	HMR
Train	6	1.5	2.9	0.03	0.40	0.31
Test	6	0.25	0.66	0.008	0.08	0.19
Test	1	0.18	0.25	0.007	0.07	0.09

TABLE 2.5: Processing time (in sec.) of 1 step. 2D SHN is the default stack hourglass network for 2D joints estimation. Both 2D and vol. SHN have 5 stacks with 41 heatmaps/points. Vol. SHN has 16 depth bins. SMPLR includes SMPL on top. HMR (Kanazawa et al., 2018) uses ResNet-50-V2 as the CNN backbone which is 3× faster than vol. SHN in test time for a batch size of 1.

increased less than 1mm in average which is quite marginal on this dataset. Fig. 2.6 shows some qualitative results.

SURREAL. The competitors on this dataset are (Varol et al., 2018) and (Tung et al., 2017). Note that (Varol et al., 2018) relies on all ground truth data in a multi-task setup to generate volumetric body shape which limits applicability. That is, volumetric body shape is a coarse representation and not parametric. The results in Tab. 2.3 show ALL_{Proc} outperforms (Varol et al., 2018) by 3.5mm for SMPL surface error. Interestingly, our SHN_{e2e}^{final} achieves same 3D joints estimation error (40.8 mm) than (Varol et al., 2018) without performing multi-task learning. We show some qualitative results in Fig. 2.6.

UP-3D. We use this dataset to show results in a in-the-wild scenario lifting 2D joints to 3D. To do so, we fine-tune SHN_{e2e}^{final} pre-trained on SURREAL. Note that we do not train this model end-to-end on UP-3D dataset. Since this dataset is of small size, we include a subset of 18K images from SURREAL for training. During training the input to DAE is ground truth joints plus uniform noise and depth is set to 0 following the same procedure as (Martinez et al., 2017). During testing, SHN estimations (with depth set to 0) are fed to DAE. The inputs to SMPLR are DAE estimations. The SMPL errors in test set before and after Procrustes mapping are around 91mm and 87mm, respectively, being below the 100.5mm error reported in (Pavlakos et al., 2018). Fig. 2.7 shows some qualitative results.

Silhouette IoU. To check the quality of rendered body, we also compute silhouette IoU of the estimated mesh after projection to image plane. The results can be seen in Tab. 2.4. We achieve a high IoU (more than 0.7) on all datasets without explicitly training the network for this task.

2.4.7 Time complexity

We show the processing time of each module of the proposed network in Tab. 2.5. Experiments were conducted on a GTX1080TI GPU. We compare a default 2D SHN with our proposed volumetric SHN (both SHNs have 5 stacks). As expected, volumetric SHN is around 2 times slower than 2D SHN in the training for a batch size of 6. However, there is not much difference between them at inference time for a batch size of 1. We must note that 2D SHN can be fit in GPU memory for a batch size of 12 while volumetric SHN can have at most a batch size of 6. Our SMPLR implementation can be run in 14 FPS for a batch size of 1 at inference time. As one can see, the CNN backbone is the most time-consuming module. We also run HMR (Kanazawa et al., 2018) which uses ResNet-50-V2 as the CNN backbone and compare in Tab. 2.5. As expected ResNet-50-V2 runs 3× faster than volumetric SHN in test time for a

batch size of 1. We note that our approach is not dependent on CNN backbone and volumetric SHN can be replaced by any state-of-the-art 3D joints estimation CNN.

2.5 Conclusions

We proposed a deep-based framework to recover 3D pose and shape from a still RGB image. Our model is composed of a SHN backbone followed by a DAE and a network capable of reversing SMPL from sparse data. Such model is able to accurately reconstruct human body mesh benefiting from end-to-end training. We experimentally found that processing SHN output joints with DAE removes structured error. We have also shown that SMPL model can be reversed and used to recover 3D pose and shape. Finally, we exploit SMPLR capabilities in the training of deep learning networks by backpropagating SMPL related errors through the SHN. We evaluated our proposal on SURREAL and Human3.6M datasets and improved SMPL-based state-of-the-art alternatives by 3.5 and 12 mm, respectively on each dataset.

Our model is flexible in design and applicability. For instance, SHN can be replaced by shallow models in mobile devices while DAE helps to improve performance. In the lack of 3D annotations for RGB images, estimated 2D joints can be lifted to 3D by training DAE on synthetic data. However, end-to-end training is not applicable in this case. Also, two independent networks for pose and shape regression make SMPLR explainable w.r.t. these parameters. Although landmarks help to disambiguate joint orientation, annotating them in in-the-wild datasets is a challenging task.

As future work, hybrid approaches could be applied to improve joint orientation. Given the difficulty of data annotation, unsupervised training is also a demand. At the end, accurate body pose and shape estimation under variable clothing has not been vastly explored due to cloth simplicity in current datasets. This is also an interesting future work in the field. Moreover, not only human pose and shape should be regressed, but also the 3D garments. To do so, it is first required a deep understanding of 3D cloth. This line of research is explored from Chapter 3 onward. The first problem to tackle is the lack of publicly available datasets with challenging garments that present realistic cloth dynamics.

Chapter 3

CLOTH3D: Clothed 3D Humans

FIGURE 3.1: Left: CLOTH3D is the first big scale datasets of animated clothed humans. It contains thousands of different outfits and subjects, high variability of poses and rich cloth dynamics. Right: generated 3D garments with proposed GCVAE. (`http://chalearn.cvc.uab.es/dataset/38/description/`)

3.1 Introduction

Previously, on Chapter 2, we discussed about human 3D pose and shape recovery from RGB. As a next step, it is useful to tackle also garment regression from RGB. Nonetheless, 3D clothing is a challenging problem with many sides to be studied to achieve a fine understand of garments. The modelling, recovery and generation of 3D clothes will allow for enhanced virtual try-ons experience, reducing designers and animators workload, or understanding of physics simulations through deep learning, just to mention a few applications. However, current literature in the modelling, recovery and generation of clothes is almost focused on 2D data (Dong et al., 2017; Lin et al., 2015; Pumarola et al., 2019b; Shin et al., 2019). This is because of two factors. First, deep learning approaches are data-hungry, and nowadays not enough 3D cloth data is available (see Tab. 3.1). Second, garments present a huge variability in terms of shape, sizes, topologies, fabrics, or textures, among others, increasing the complexity of representative 3D garment generation.

One could define three main strategies in order to produce data of 3D dressed humans: 3D scans, 3D-from-RGB, and synthetic generation. In the case of 3D scans, they are costly, and at most they can produce a single mesh (human + garments). Alternatively, datasets that infer 3D geometry of clothes from RGB images are inaccurate and cannot properly model cloth dynamics. Finally, synthetic data is easy to generate and is ground truth error free. Synthetic data has proved to be helpful to train deep learning models to be used in real applications (Nikolenko, 2019; Ros et al., 2016; Varol et al., 2017b).

Dataset	3DPW (Marcard et al., 2018)	BUFF (Zhang et al., 2017)	Untitled (Wang et al., 2018)	3DPeople (Pumarola et al., 2019a)	TailorNet (Patel et al., 2020)	CLOTH3D
Resolution	2.5cm	0.4cm	1cm	-[1]	1cm	1cm
Missing	x	✓	x	x	x	x
Dynamics	x	✓	x	x	x	✓
Garments	18[2]	10∼20	3[3]	High[4]	20	11.3K
Fabrics	x	x	x	x	x	✓
Poses[5]	Low	Low	Very low	Low	1782	High
Subjects	18[2]	6	2K	80	9	8.5K
Layered	x	x	✓	-[1]	✓	✓
#samples	51k	11K	24K	2.5M	55.8k	2.1M
Type	Real	Real	Synth.	Synth.	Synth.	Synth.
RGB	✓	x	✓	✓	x	x
GT error	26mm	1.5-3mm	None	None	None	None

TABLE 3.1: CLOTH3D vs. available 3D cloth datasets. [1]: 3D data includes depth, normal and scene flow maps, but not 3D models. [2]: 3DPW contains 18 clothed models that can be shaped as SMPL. [3]: garments of (Wang et al., 2018) are shaped to different sizes. [4]: Garment variability not specified, nonetheless, authors propose a generation pipeline that can modify template garments into many different sizes. [5]: poses are strongly related to number of frames, and in (Wang et al., 2018) most samples share the same static pose.

In this work, we present CLOTH3D, the first synthetic dataset composed of thousands of sequences of humans dressed with high resolution 3D clothes, see Fig.3.1. CLOTH3D is unique in terms of garment, shape, and pose variability, including more than 2 million 3D samples. We developed a generation pipeline that creates a unique outfit for each sequence in terms of garment type, topology, shape, size, tightness and fabric. While other datasets contain just a few different garments, ours has thousands of different ones. On Tab. 3.1 we summarize features of existing datasets and CLOTH3D.

Additionally, we provide a baseline model able to generate dressed human models. Similar to (Alldieck et al., 2018a; Ma et al., 2019; Yang et al., 2018a) we encode garments as offsets connecting skin to cloth, using SMPL(Loper et al., 2015b) as human body model. This yields an homogeneous dimensionality on the data. As in (Pons-Moll et al., 2017), we use a segmentation mask to extract the garment by removing body vertices. In our case, the mask is predicted by the network. We propose a Conditional Variational Auto-Encoder (CVAE) based on graph convolutions (Bronstein et al., 2017; Defferrard et al., 2016; Ma et al., 2019; Niepert et al., 2016; Wu et al., 2019; Yuan et al., 2019) (GCVAE) to learn garment latent spaces. This later allows for the generation of 3D garments on top of SMPL model for any pose and shape (right on Fig.3.1).

3.2 Related Work

3D Garment Datasets. Current literature on 3D garment lacks on large public available datasets. One strategy to capture 3D data is through **3D scans**. The BUFF dataset (Zhang et al., 2017) provides high resolution 3D scans, but few number of subjects, poses and garments. Furthermore, scanning techniques cannot provide layered models (one mesh for the body and one for each garment) and often one can find regions occluded at scanning time, meaning missing vertices or corrupted shapes. The work of (Pons-Moll et al., 2017) proposed a methodology to segment

scans to obtain layered models. Authors of (Yu et al., 2019) combined 3D scans with cloth simulation fitting at each frame to deal with missing vertices. Similarly, (Bhatnagar et al., 2019) provided a dataset from 3D scans. However, the amount of samples is in the order of a few hundreds. The 3DPW dataset (Marcard et al., 2018) is not focused on garments, but rather on pose and shape in-the-wild. The authors proposed a modified SMPL **parameterized model** for each outfit (18 clothed models), which, as SMPL, can be shaped and posed. Nevertheless, resolution is low and posing is through rigid rotations. Therefore, cloth dynamics are not represented. The dataset of (Wang et al., 2018) is synthetically created through **physics simulation**, with three different garment types: tshirt, skirt and kimono. They propose an automatic garment resizing based on real patterns, but provide only static samples on few different poses. The work of (Patel et al., 2020) also includes a synthetic dataset obtained through simulation of 20 combinations of different garment styles and body shapes into 1782 static poses. Finally, 3DPeople dataset (Pumarola et al., 2019a) is the most comparable to ours in terms of scale, but has significant differences w.r.t. CLOTH3D. On one hand, this dataset has been designed specifically for computer vision. Data are given as **multi-view images** (RGB, depth, normal and scene flow), there are no 3D models. On the other hand, the garments are rigged models, so there is no proper cloth dynamics. And lastly, source pose data is sparse, 70 pose sequences with an average length of 110 frames. Our CLOTH3D dataset aims to overcome previous datasets issues. We automatically generate garments to obtain a huge variability on garment type, topology, shape, size, tightness and fabric. Afterwards, we simulate clothes on top of thousands of different pose sequences and body shapes. Tab.3.1 shows a comparison of features for existing datasets and ours. In CLOTH3D we focus on sample variability (garments, poses, shapes), containing realistic cloth dynamics. 3DPW and 3DPeople sequences are based on rotations on rigged models, datasets of (Patel et al., 2020; Wang et al., 2018) contain static poses only, and BUFF has very few and short sequences. Moreover, none other provides metadata about fabrics, which has a strong influence on cloth behaviour. Similarly, the scarcity of these datasets implies low variability on garments, poses and subjects. Finally, note how only synthetic datasets provide with layered models and have no annotation error.

3D Garment Generation. Current works in 3D clothing focus on the generation of dressed humans. We split related work into non-deep and deep-learning approaches. Regarding **non-deep learning**, the authors of (Guan et al., 2012) proposed a data-driven model that learns deformations from template garment to garment fitted to the human body, shaped and posed. They factorize deformations into shape-dependant and pose-dependant by training on rest pose data first, and later on posed bodies. Transformations are learnt per triangle, and thus it yields inconsistent meshes that need to be reconstructed. The data-driven model of (Pons-Moll et al., 2017) is able to recover and retarget garments from 4D scan sequences relying on masks to separate body and cloth. Authors propose an energy optimization process to identify underlying body shape and garment geometry, later, cloth displacements w.r.t. body are computed and applied to new body shapes. This means information such as wrinkles is "copied" to new bodies, which produces valid samples but cannot properly generate its variability. Regarding **deep learning** strategies, the work of (Gundogdu et al., 2019a) deals with body and garments as different point clouds through different streams of a network, which are later fused. They also use skin-cloth correspondences for computing local-features and losses through nearest neighbour. The works of (Alldieck et al., 2018a; Ma et al., 2019; Patel et al., 2020; Yang et al., 2018a) consider encoding clothes as offsets from SMPL body model with

different goals. In (Ma et al., 2019) authors propose a combination of graph VAE and GAN to model SMPL offsets into clothing. Similarly, in (Patel et al., 2020), authors propose encoding garments as SMPL offsets and topology as a subset of SMPL vertices, later, they learn two models for low and high frequency details which effectively generate realistic wrinkles on the garments. In (Wang et al., 2018; Yang et al., 2018a) a PCA decomposition is used to reduce clothing space. In (Alldieck et al., 2019; Lahner et al., 2018), authors register garments to low resolution meshes (garment templates and SMPL respectively), to later use UV normal maps to represent high-frequency cloth details (wrinkles). Authors of (Santesteban et al., 2019) propose learning Pose Space Deformation models for template garments by training deep models instead of SVD (as SMPL). The work of (Wang et al., 2019) presents a template garment autoencoder where latent spaces are disentangled into motion and static properties to realistically interpolate into 3D keyframes. Similar to previous approaches, our proposed methodology also encodes clothes as SMPL offsets. Nevertheless, the assumption that garments follow body topology does not hold for skirts and dresses. In this sense, we propose a novel body topology specific for those cases. Additionally, our model predicts garment mask along offsets to generate layered models.

3.3 Dataset

CLOTH3D is the first big scale dataset of 3D clothed humans. The dataset is composed of 3D sequences of animated human bodies wearing different garments. Fig. 3.1 depicts a sequence (first row) and randomly sampled frames from different sequences. Samples are layered, meaning each garment and body are represented by different 3D meshes. Garments are automatically generated for each sequence with randomized shape, tightness, topology and fabric, and resized to target human shape. This process yields a unique outfit for each sequence. It contains over 7000 non-overlapping sequences of 300 frames each at 30fps, yielding a total of 2.1M samples. As seen in Tab. 3.1, garment and pose variability is scarce in available datasets, and CLOTH3D aims to fill that gap. To ensure garment type balance, given that females present higher garment variability, we balance gender as 2:1 (female:male). Finally, for validation purposes, we split the data in 80% sequences as training and 20% as test. Splitting by sequences ensures no garment, shape or pose is repeated in training and test.

The data generation pipeline starts with sequences of human bodies in 3D. Human pose data source is (*Carnegie-Mellon Mocap Database* n.d.), later transformed to volumetric bodies through SMPL (Loper et al., 2015b). These sequences might present body self-collisions which will hinder cloth simulation, not only on affected regions, but also in global garment dynamics. We automatically solve collisions or reject these samples. Human generation process is described in subsec. 3.3.1. Later, we generate unique outfits for each sequence. We start from a few template meshes which are randomly shaped, cut and resized to generate a unique pair of garments for each sample, with the possibility to be combined into a single full-body garment. Fig. 3.2 shows the generation process, which is also detailed in subsec. 3.3.2. Finally, once human sequence and outfit are done, we use a physics based simulation to obtain the garment 3D sequences. Simulation details are described in subsec. 3.3.3.

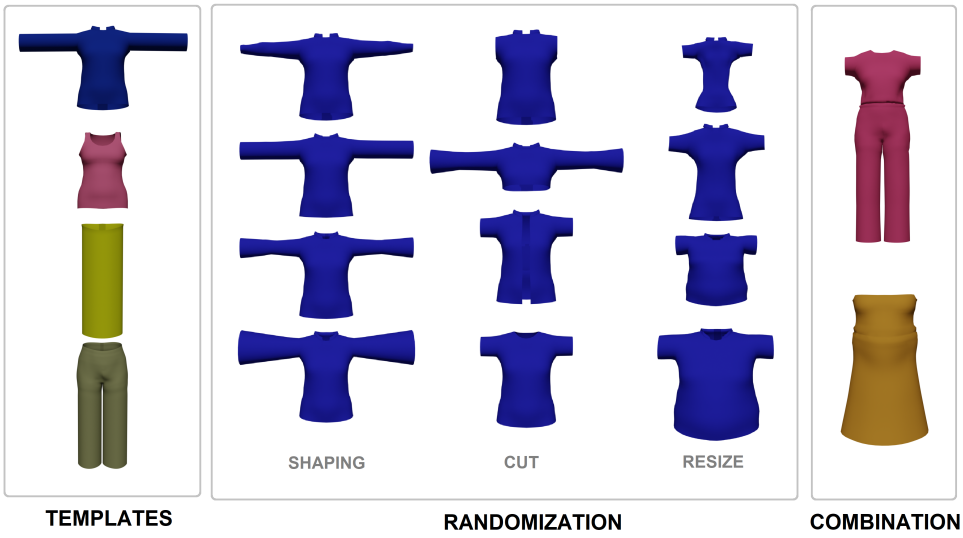


FIGURE 3.2: Unique outfit generation pipeline. First, one upper-body and lower-body garment template is selected. Then, garments are individually shaped, cut and resized. Finally, garments might be combined into a single one.

3.3.1 Human 3D Sequences

SMPL. It is a parametric human body model which takes as input shape $\beta \in \mathbb{R}^{10}$ and pose $\theta \in \mathbb{R}^{24\times3}$ to generate the corresponding mesh with 6890 vertices. We use this model to generate animated human 3D sequences. We refer to (Loper et al., 2015a) for SMPL details. To generate animated bodies, we need a source of valid sequences of SMPL pose parameters $\theta \in \mathbb{R}^{f\times24\times3}$. We take such data from the work of (Varol et al., 2017b), where pose is inferred from CMU MoCap data (*Carnegie-Mellon Mocap Database* n.d.) following the methodology proposed at (Loper et al., 2014). These pose data come from around 2600 sequences of 23 different actions (dancing, playing, running, walking, jumping, climbing, etc.) performed by over 100 different subjects. SMPL shape deformations are linearly modeled through PCA. To obtain a balanced dataset we uniformly sample shape within range $[-3, 3]$ for each sequence.

Self-collision. Body collides with itself for certain combinations of pose and shape parameters. Intersection volumes create regions where simulated repel forces are inconsistent, corrupting global cloth dynamics. We classify these collisions in three generic cases. Solvable Fig.3.3(a): small intersection volumes near joints, specially armpits and crotch. We use SMPL body parts segmentation to linearly separate the collided vertices to permit a correct simulation. Separation space is 4mm so that a folded cloth can fit. Unsolvable Fig.3.3(b): big intersection volumes or incompatible intersections (e.g.: arm vs. leg). We reject or re-simulate with thinner body. Special cases Fig.3.3(c): removing hands, forearms or arms for short-sleeved upper-body and lower-body garments significantly increases the amount of valid samples. This requires manual supervision. Self-collision solution is not stored, hence, if collided vertices change significantly, garments might present interpenetration w.r.t. unsolved body. Only small intersected volumes are corrected and the rest are rejected (or simulated with thinner body). The goal of self-collision solving is to avoid invalid cloth dynamics. Accurate, realistic solving of soft-body self-collision is out of the scope of this work.

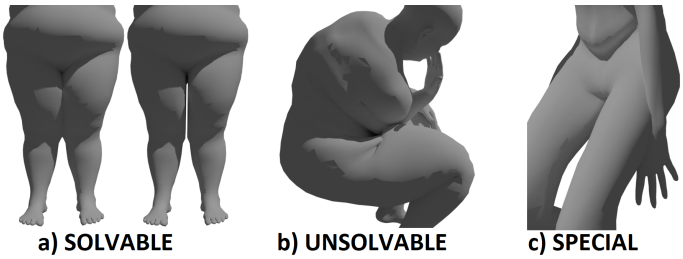


FIGURE 3.3: Types of self-collision: a) collided vertices can be linearly separated with the aid of a body part segmentation, b) no trivial solution, we reject this kind of sample, c) correct simulation might be possible if forearm is removed.

3.3.2 Garment Generation

Garment Templates. Generation starts with a few template garments for each gender. Garments can be classified in upper-body and lower-body. Lower-body can be further split into trousers and skirts. These three categories, and combinations between them, encompass almost any day-to-day garment. Template garments have been manually created by designers from real patterns and are: t-shirt, top, trousers and skirt.

Shaping. On sleeves, legs and skirt, we find a significant shape variability. It is possible to define them as cylinders of variable width around certain axes: along arms for sleeves, legs for trousers and vertical body axis for skirt. For sleeves and legs, width will be constant or decreasing while moving towards wrist/ankle, and beyond a randomly sampled point along its axis, it might start increasing (widening). For skirts, width always increases, from waist to bottom. Rate of width decrease/increase is uniformly sampled within ranges empirically set per garment. More formally:

$$W(x) = \alpha_1 x + \alpha_2 \max(0, x - x_{\text{offset}}) + W_0, \tag{3.1}$$

where x is position along axis (0 at shoulder/hips), $W(x)$ is width at position x, W_0 is width at $x = 0$, x_{offset} is a uniformly sampled point along the axis and α_1 and α_2 are constants empirically defined for each garment. For t-shirts and trousers, $\alpha_1 < 0 < \alpha_2$. For skirts, $\alpha_1 > \alpha_2 = 0$.

Cut. Template garments cover most of the body (long sleeves, legs and skirt). At this generation step, garments are cut to increase variability on length and topology. Cuts are along arms, legs and torso. Plus, upper-body garments have specific cuts to generate different types of garments (e.g., t-shirt, shirt, polo).

Resizing. Garments are resized to random body shapes. It is safe to assume that size variability on garments is similar to body shape variability. Following this reasoning, SMPL shape displacements are transferred to garments by nearest neighbour. Nevertheless, this process is noisy and human body details are transferred to garment. To address these issues, an iterative Laplacian smoothing is applied to shape displacements, removing noise and filtering high frequency body details, while preserving the geometry of the original garment. On SMPL, first and second shape parameters correspond to global human size and overall fatness. Knowing this, garments are resized to a different target shape. This new shape has two offsets at first and second parameters, the garment tightness $\gamma \in \mathbb{R}^2$. These offsets on garment resizing will generate loose or tight variability. As tighter garments present less dynamics and complexity, we bias the generator towards loose clothes by sampling tightness on the range $[-1.5, 0.5]$.

Walk	Animal	Fight	Jump	Run	Sing	Wait	Swim	Story	Sports	Dance	Yoga	Spin
27.49%	10.79%	4.38%	2.78%	2.49%	2.38%	2.31%	1.97%	1.70%	1.63%	1.37%	1.01%	0.90%

Exercise	Climb	Carry	Stand	Wash	Balancing	Trick	Sit	Interact	Drink	Pose	Others
0.84%	0.71%	0.67%	0.66%	0.63%	0.54%	0.51%	0.28%	0.20%	0.14%	0.14%	33.48%

TABLE 3.2: CLOTH3D statistics per action label.

Jumpsuits and Dresses. Full-body garments can be generated by combining upper-body and lower-body garments. After generating the clothes individually, a final step automatically sews them together.

3.3.3 Simulation

Cloth simulation is performed on Blender, an open source 3D creation suite. Blender's cloth physics, as it is in version 2.8, has been implemented with state-of-the-art algorithms based on mass-spring model. The simulation performs $420 - 600$ steps per second, depending on the complexity of the garment.

Fabrics. Changing the parameters of the mass-spring model allows simulation of different fabrics. Blender provides different presets for *cotton*, *leather*, *silk* and *denim*, among others. These four fabrics have been used for the creation of the dataset. Upper-body garments might be cotton or silk, while the rest of the garment types can be any of those fabrics. Different fabrics produce different dynamics and wrinkles on simulation time.

Elastics. At simulation time, sleeves and legs have a 50% chance each of presenting an elastic behaviour at their ends, also at waist on full-body garments.

3.3.4 Additional dataset statistics

Tab.3.2 shows the CLOTH3D statistics in terms of action labels by grouping them into generic categories. Note that original data action label is very heterogeneous, specific and incomplete. These labels are gathered from CMU MoCap dataset. We observe a high density on *Walk*, but it is important to note that this gathers many different sub-actions (walk backwards, zombie walk, walk stealthily, ...) as many other action labels do. Additionally, most of these actions were performed by different subjects, which implies an increase in intra-class variability. The label 'others' contains all action labels that cannot be included in any of the categories plus all the missing action labels.

3.4 Dressed Human Generation

This section presents the methodology for deep garment generation. As (Alldieck et al., 2018a; Ma et al., 2019; Patel et al., 2020; Yang et al., 2018a), data dimensionality and topology is fixed by encoding it as body offsets. In addition, by masking body vertices we represent different garment types and separate them from the body, e.g. in a similar fashion to (Patel et al., 2020; Pons-Moll et al., 2017). To compute ground truth offsets, a body-to-garment matching is needed. A dedicated algorithm for this task should be able to correctly register skirt-like garments which have a different topology than the body. In sec. 3.4.1 we explain details of our data pre-processing. Our proposed model is a Graph Conditional Variational Auto-Encoder (GCVAE).

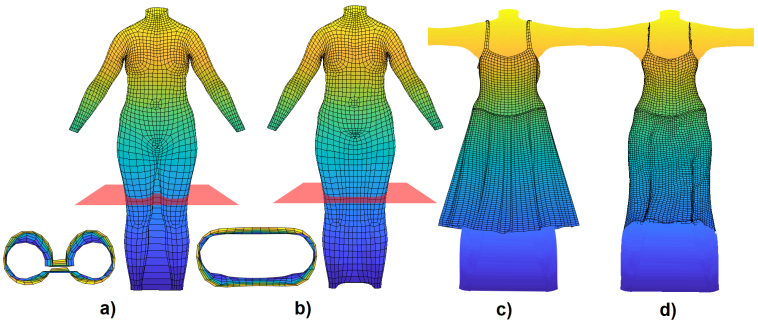


FIGURE 3.4: Dual topology and registration. a) New additional proposed topology, where inner legs are connected. This topology is used for graph convolutions as well. b) Result of Laplacian smoothing of inner leg vertices. It is used only for skirt/dress registration. We show top view of meshes around an imaginary red cutting plane. c) Garment in rest pose. d) Garment registered to body model.

By conditioning on available metadata (pose, shape and tightness), we learn a latent space encoding specific information about garment type and its dynamics (details are given in sec. 3.4.3). Fig. 3.5 illustrates the proposed model.

3.4.1 Data Pre-processing

In order to match among garment and body, we apply non-rigid ICP (Amberg et al., 2007). Registration is performed once per sequence in rest pose. Due to SMPL low vertex resolution, garment details could be lost. For this reason we subdivide the mesh (and corresponding SMPL model parameters). Head, hands and feet are not used to find correspondences and removing them halves input dimensionality. This yields a final mesh with $N = 14475$ vertices. Finally, note that skirt-like garments do not follow the same topology as SMPL mesh. For this task we introduce a novel topology explained on the subsection below. An example of the registration is shown in Fig. 3.4. Finally, body to cloth correspondences and garment mask are extracted by nearest neighbor matching.

3.4.2 SMPL-Skirt Topology

From SMPL body mesh, a 'column' of inner faces of each leg is removed and a new set of faces is created by connecting vertices from both legs, see Fig.4a. New faces are highly stretched, producing noisy garment registrations if used as is, NR-ICP yields optimal results for homogeneous meshes (in garment domain). Because of this, we apply an iterative Laplacian smoothing to vertices belonging to the inner parts of each leg, see Fig.4b for the result. This process is repeated before registration with the corresponding shape of the subject in the sequence in T-pose. This gives a matching between garment and body vertices to compute offsets. For encoding garments as offsets we use body mesh without smoothing, as this process will misbehave for posed bodies. Finally, for graph convolutions, we use the Laplacian matrix corresponding to this new topology for garments of type Dress and Skirt. This ensures that vertex deep features are aggregated with the correct neighbourhood. Afterwards, we transfer body topology to the predicted garment, and it is therefore crucial to use the correct topology for each garment type.

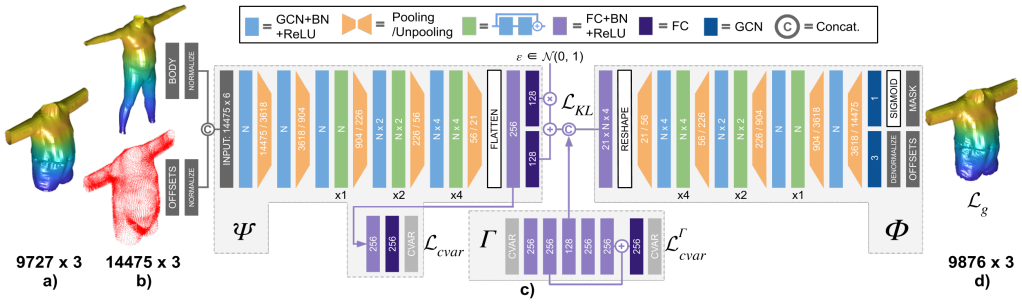


FIGURE 3.5: Model pipeline. a) Input garment b) body and offsets w.r.t. body (Sec. 3.4.1). Model input is the concatenation of body and offsets. c) Network architecture. Conditional variables (CVAR) are processed by an AutoEncoder. To improve latent space factorization, CVAR are also regressed from the first encoder FC layer. Decoder outputs are offsets and mask. d) Reconstruction of the garment by adding offsets to body and removing body vertices according to mask. We set N as 128.

3.4.3 Network

As shown in Fig. 3.5, our network is based on a VAE generative model. The goal is to learn a meaningful latent space associated to the garments of any type, shape or with wrinkles which is used to generate realistic draped garments. Garment type and shape are associated to the static state of the garment while wrinkles belong to the dynamics of the garments. Here, we disentangle the latent space between statics and dynamics of the garments, and refer to learnt latent codes as garment code ($z_s \in \mathbb{R}^{128}$) and wrinkle code ($z_d \in \mathbb{R}^{128}$), respectively. To do so, we build two separate networks, one trained on static garments (so called SVAE) and one trained on dynamic garments (so called DVAE). To factorize the latent space from irrelevant parameters to the garment type and shape, we condition SVAE on body shape ($\beta \in \mathbb{R}^{11}$)[1] and garment tightness ($\gamma \in \mathbb{R}^2$). Likewise, DVAE is conditioned on β, γ, body pose ($\theta \in \mathbb{R}^{f\times 72}$) and z_s, where f is the number of frames in a temporal sequence. Let $cvar_s$ and $cvar_d$ be the stacking of conditioning variables of SVAE and DVAE in a single vector. It is worth noting that θ is constant in SVAE so that we do not include it in $cvar_s$. We implement graph convolutions as in (Bronstein et al., 2017; Defferrard et al., 2016; Ma et al., 2019; Niepert et al., 2016; Wu et al., 2019; Yuan et al., 2019). We also include skip connections throughout the whole network.

Architecture. Let $X_s \in \mathbb{R}^{V_T\times 3}$ and $X_d \in \mathbb{R}^{V_T\times 3}$ be offsets computed on static and dynamic samples, respectively. From now on we use subscript s and d for static and dynamic variables and discard them for general cases. SVAE and DVAE have a similar structure with three main modules: encoder $\{cvar^z, z\} = \Psi(\bar{X}, \bar{T})$, conditioning $\{cvar, cvar^z\} = \Gamma(cvar)$ and decoder $\{\bar{X}, M\} = \Phi(z, cvar^z)$, where $M \in \mathbb{R}^{V_T\times 1}$ is the garment mask. Conditioning network Γ is an autoencoder with one skip connection and $cvar^z$ is its middle layer features. The goal of this network is to provide a trade-off between $cvar$ and z. The architecture details are shown in Fig. 3.5. Note that all GCN layer features (except first and last layers) are doubled in DVAE vs. SVAE. We refer the reader to the supplementary material for more details on the network architecture.

Pooling. We resort to a mesh simplification algorithm (Garland et al., 1997) to create a hierarchy of meshes with decreasing detail in order to implement the pooling operator. We follow (Yuan et al., 2019) to have vertices uniformly distributed in the graph coarsening. However, this approach does not guarantee a uniform or

[1] We include gender as an additional dimension to the shape parameters.

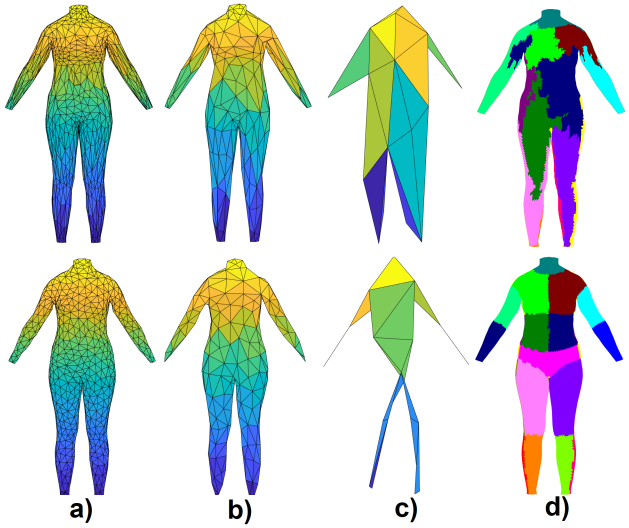


FIGURE 3.6: Mesh hierarchy for pooling. Upper: default (Garland et al., 1997). Lower: proposed. a), b) and c) depict the mesh hierarchy used for graph pooling through the model. Observe the difference on spatial distribution at a) and b). c) shows how lowest pooling is more meaningful regarding the segments (one vertex per segment). d) is the visualization of correspondences (receptive field) between highest and lowest hierarchy levels. The proposed pooling yields more meaningful pooling receptive fields w.r.t. body parts.

meaningful receptive field on a high resolution mesh. To achieve a homogeneous distribution of correspondences throughout the body between pooling layers, we define a segmentation (Fig. 3.6(d)) and forbid the algorithm from contracting edges connecting vertices of different segments. Segmentation contains 21 segments and it is designed such that regions of the body with highest offset variability have smaller segments. Thus, more capacity of the network is available to model those parts. See Fig. 3.6. Our mesh hierarchy is formed by 6 different levels. The dimensionality of those meshes is: $14475 \rightarrow 3618 \rightarrow 904 \rightarrow 226 \rightarrow 56 \rightarrow 21$, leaving a single node for each segment on the last pooling layer. We use max-pooling in the proposed hierarchy. For unpooling, features are copied to all corresponding vertices of the immediate higher mesh.

Loss. We train conditioning network Γ independently using $L1$ loss and freeze its weights while training VAE. S/DVAE loss is a combination of a garment related term, a *cvar* term and KL-divergence:

$$\mathcal{L} = \mathcal{L}_g + \mathcal{L}_{cvar} + \lambda_{KL} D_{KL}(q(z|X, cvar)||p(z|cvar)), \tag{3.2}$$

Garment related term handles offsets, mask (if available), smoothness and collisions:

$$\mathcal{L}_g = \mathcal{L}_o + \lambda_n \mathcal{L}_n + \lambda_m \mathcal{L}_m + \lambda_c \mathcal{L}_c, \tag{3.3}$$

where $\mathcal{L}_o$ is an L1-norm applied to output offsets. $\mathcal{L}_n$ is the smoothness term based on L1-norm on normals. We found that regular Laplacian loss ensures smoothness at the cost of losing high frequency geometric details, while a normal loss makes output geometry consistent w.r.t. the input. $\mathcal{L}_m$ consists on L1-norm on mask. Finally, $\mathcal{L}_c$ is the collision loss. Given that garments are represented as offsets, we design this loss as:

$$\mathcal{L}_c = max(0, -o \cdot V_N), \tag{3.4}$$

where o are the output offsets and V_N are the body normals at the corresponding

	Surface	Normals	Mask	KL loss
All	14.3	1.04	0.9518	0.9820
No normals	22.8	1.07	0.9472	0.5966
No mask	92.7	1.19	-	0.8799
No collision	14.7	1.02	0.9522	0.9414
No CVAR	14.8	1.02	0.9520	1.1009
Default pooling	14.9	1.03	0.9390	0.7623

(A)

	Surface	Normals	Mask	KL loss
Top	11.9	1.20	0.9035	0.9536
T-shirt	15.5	1.21	0.9565	1.1701
Trousers	10.9	0.84	0.9475	0.9008
Skirt	21.4	0.79	0.9520	1.0255
Jumpsuit	13.3	1.07	0.9637	0.8788
Dress	16.7	1.06	0.9662	0.9995

(B)

(C) Table 2: (a) Ablation results on the static dataset for all clothes. (b) Ablation results (full model) on the static dataset for each cloth category. Surface and normal errors are shown in mm and radians, respectively.

# frames	Top	T-shirt	Trousers	Skirt	Jumpsuit	Dress	Avg.
1	21.8/1.24	28.8/1.29	20.7/0.89	37.6/0.92	28.2/1.15	35.5/1.13	29.0/1.10
4	20.1/1.23	28.0/1.28	18.5/0.86	33.2/0.89	26.1/1.09	32.2/1.11	26.1/1.08

TABLE 3.4: Ablation results (full model) on the dynamic dataset conditioning on different number of frames. Left: surface error (mm) / Right: normals error (radians).

vertices, this penalizes offsets that go within the body. $\mathcal{L}_{cvar}$ is $L1$ loss on encoder $cvar^z$ regressor.

3.5 Experiments

First, we detail the metrics chosen to analyze the results.

Surface. Given that input and prediction have the same dimensionality and order, we use standard euclidean norm (in mm.).

Normals. Measure of surface quality. We compute normals error based on mesh face normals by their angle difference (in radians) to ground truth normals.

Mask. Garment mask is evaluated by the intersection over union (IoU).

KL Loss. We use KL loss as a measure of quality of latent code factorization and meaningfulness of the latent space.

3.5.1 Ablation Study

We trained SVAE on an additional dataset of static samples (in rest pose) with 30K samples. 20% of the data is kept for evaluation and the rest for training. The results are shown in Tab. 3.3a and 3.3b.

Normals. Looking at the second row of Tab. 3.3a we observe that enforcing a reconstruction consistent with normals significantly reduces surface error and, as expected normals error. However, including normals has a negative impact on KL loss comparing to first row.

Mask. As seen in third row of Tab. 3.3a, both, surface and normals error are significantly higher without mask prediction (comparing to first row).

Collision. Fourth row of Tab. 3.3a shows how collision loss helps to improve vertex location by pushing collided vertices to their correct position. On the other hand, it is observable a non-significant increase on other losses.

CVARs. As explained in Sec. 3.4.3, conditional variables are regressed from the first FC layer of the encoder to improve latent space factorization. On fifth row of

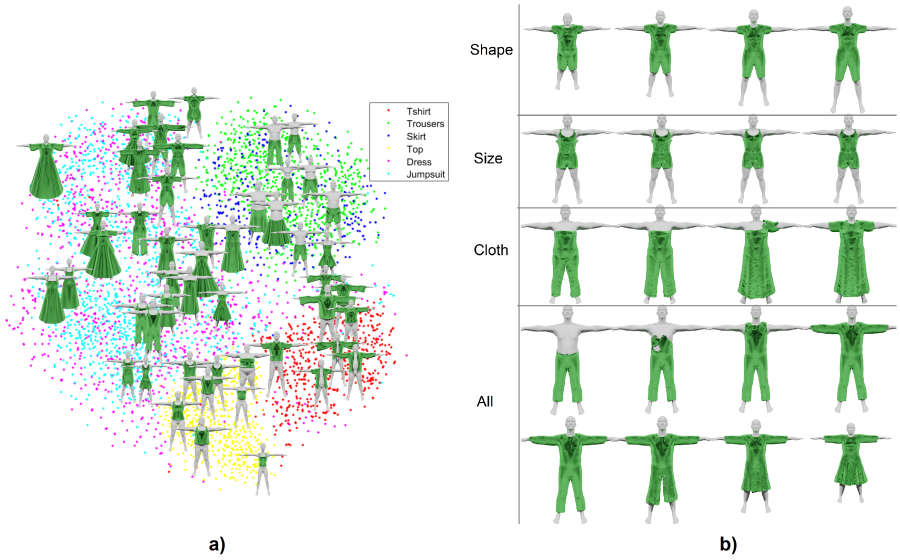


FIGURE 3.7: a) Visualization of the learned latent space for static samples using t-SNE algorithm. b) Transitions of static samples. First three rows: conditioning on shape, tightness or cloth while the rest are fixed. Last two rows: transition of all variables. Variables are linearly graduated.

Tab. 3.3a we can see that, while surface or normals error have no significant differences, KL loss improves.

Pooling. On Sec. 3.4.3 we discussed different approaches for tackling the pooling on a graph neural network. To do this, we built a mesh hierarchy. We compared default mesh simplification algorithm versus our proposed modification. Results are shown in the last row of Tab. 3.3a. While improvement on surface and normals errors is marginal, this new pooling benefits mask prediction.

Per Garment Category Error. Results per garment are shown in Tab. 3.3b. Skirts present the highest surface error, as its vertices are further away from the body compared to other garments. Following this reasoning, we find trousers having the less surface error. If we look at normals error, we find an opposite behaviour for skirts, as their geometry is the simplest one. On the other hand we see that upper-body garments present more complex geometries, and therefore, higher normals error. Looking at mask error, we see that garments that cover most of the body have the lowest error. This is due to IoU metric nature, the lower the number of points, the more impact shall have each wrong prediction. Finally, looking at KL loss, we observe the model has difficulties to obtain meaningful spaces for T-shirts. As explained on Sec. 3.3.2, T-shirts category includes open shirts as well, which highly increases class variability. We also see that trousers and jumpsuits have the lowest KL loss.

Learned Latent Space. In Fig. 3.7, we show distribution of 5K random static samples computed by t-SNE algorithm. As one can see, the proposed GCVAE network can group garments in a meaningful space. Interestingly, dress and jumpsuit that share more vertices also share the same latent space. Additionally, we show garment transitions in this space in Fig. 3.7. One can see how garments transit between two different topologies (3rd row) or among different genders and shapes (4th row).

We study DVAE model in Tab. 3.4. We condition DVAE on pose for a single frame vs. four frames. Four frames are selected every 3 frames, resulting in a 12-frame clip. Training the model on a sequence of frames leads to better results in all garment categories (3mm improvement in average). This is while we do not include

FIGURE 3.8: Garment reconstruction for sequences. Note that the model has not been trained to keep temporal consistency.

any temporal information in the encoder nor any specific sequence prediction loss. DVAE qualitative results for single frames and sequences are shown in Fig. 3.1(right) and Fig. 3.8, respectively.

3.6 Conclusions

We presented CLOTH3D, the first large scale synthetic dataset of 3D clothed humans. It has a large data variability in terms of body shape and pose, garment type, topology, shape, tightness and fabric. Generated garments also show complex dynamics, providing with a challenging corpus for 3D garment generation. We developed a baseline method using a graph convolutional network trained as a variational autoencoder, and proposed a new pooling grid. Evaluation of the proposed GCVAE on CLOTH3D showed plausible garment generation. While the proposed approach successfully learnt a common space to encode numerous garments with different topologies, sizes and shapes, there is a considerable gap on the quality of the generated predictions and the ground truth data. The encoding of garments on top of the human body is responsible for a significant loss on data quality. Nonetheless, this pre-processing is necessary to train a generative model (latent code mapped to a uniform output space). Then, in Chapter 4, the generation property is discarded and only garment animation is considered (*conditioning* to pose). This will lead to a noticeable increase on prediction quality.

Chapter 4

DeePSD: Automatic Deep Skinning And Pose Space Deformation For 3D Garment Animation

FIGURE 4.1: We present a novel approach for outfit animation. Our methodology allows generalization to unseen outfits. It can handle multiple layers of cloth, arbitrary topology and complex geometric details without retraining.

4.1 Introduction

With CLOTH3D already introduced in Chapter 3, it is now time to improve on its complementary baseline. This baseline is able to generate a huge variety of different garments, but it presents an important quality gap. The goal on this chapter is to enhance the quality of the predictions, mainly by discarding the generative part of the problem and focusing in garment animation. Virtual dressed human animation has been a topic of interest for decades due to its numerous applications in entertainment and videogame industries, and recently, in virtual and augmented reality. Depending on the application we find two main classical computer graphics approaches. On the one hand, Physically Based Simulation (PBS) (Baraff et al., 1998; Liu et al., 2017; Provot, 1997; Provot et al., 1995; Tang et al., 2013; Vassilev et al., 2001; Zeller, 2005) approaches are able to obtain highly realistic cloth dynamics at the expense of a huge computational cost. On the other hand, Linear Blend Skinning (LBS) (Kavan et al., 2008; Kavan et al., 2005; Le et al., 2012; Magnenat-thalmann et al., 1988;

Wang et al., 2007; Wang et al., 2002) and Pose Space Deformation (PSD) (Allen et al., 2002; Anguelov et al., 2005; Lewis et al., 2000; Loper et al., 2015a) models are suitable for environments with limited computational resources or real-time performance demand. To do so, realism is highly compromised. Then, computer graphics approaches present a trade-off between realism and computational performance.

Deep learning has already proven successful in complex 3D tasks (Arsalan Soltani et al., 2017; Han et al., 2017; Madadi et al., 2020; Omran et al., 2018; Qi et al., 2017; Richardson et al., 2016; Socher et al., 2012). Due to the interest in the topic and the recently available 3D datasets on garments, we see the scientific community pushing this research line (Alldieck et al., 2018b; Alldieck et al., 2019; Bertiche et al., 2020; Bhatnagar et al., 2019; Guan et al., 2012; Lahner et al., 2018; Patel et al., 2020; Santesteban et al., 2019). Most proposals are built as non-linear PSD models learnt through deep learning. These methods yield models describing one or few garment types and, therefore, they lack on generalization capabilities. To overcome this, recent works propose encoding garment types as a subset of body vertices (Bertiche et al., 2020; Patel et al., 2020). This allows generalizing to more garments, yet bounds its representation capacity to body homotopies only. Thus, these approaches need to model each garment individually and cannot handle details such as pockets nor multiple layers of cloth, heavily hurting their scalability and applicability in real life scenarios.

We propose learning a mapping from the space of template outfits to the space of animated 3D models. We will show how this allows generalization to completely unseen garments with arbitrary topology and vertex connectivity. We can achieve this by identifying edition/resizing and animation as separate tasks, and focusing on the latter. Our method works with whole outfits (instead of single garments), multiple layers of cloth and resolutions, while also allowing complex geometric details (see Fig. 4.1 for some examples). Furthermore, we achieve this with a simple and small-sized neural network. The list of our contributions is as follows:

- **Outfit Generalization.** To the best of our knowledge, our proposal is the only work able to animate completely unseen outfits without additional training. This greatly increases applicability in scenarios with ever-growing number of outfits, such as virtual try-ons and videogames, where customization is key.
- **Compatibility.** Our methodology does not predict garment vertex locations, but blend weights and blend shapes matrices. This is the standard on 3D animation, and it is therefore compatible with all graphics engines. Also, it benefits from the exhaustive optimization on animation pipelines. Pose Space Deformations are a specific case of blend shapes that are combined consistently with object pose.
- **Physical Consistency.** Related works require a final post-processing step for collision solving. Alternatively, works that train with a collision-solving loss need to find a compromise between physical constraints and vertex error. Thus, predictions still show collisions. We propose to train an independent model branch such that physical consistency losses and supervised losses do not hinder each other. This yields quasi-collision free and cloth-consistent predictions while leveraging the data as much as possible.
- **Explainability.** Mapping outfits to animated 3D models yields a more intuitive work pipeline for CGI artists. Recent works try to address garment resizing/edition along animation by encoding styles into parametric representations (Bertiche et al., 2020; Patel et al., 2020). Thus, expert knowledge is required to obtain the desired results by tuning style parameters.

4.2 State of the art

Computer Graphics. Obtaining realistic cloth behaviour is possible through **PBS** (Physically Based Simulation), commonly through the well known *mass-spring* model. Literature on the topic is extensive, focused on improving the efficiency and stability of the simulation by simplifying and/or specializing on specific setups (Baraff et al., 1998; Provot, 1997; Provot et al., 1995; Vassilev et al., 2001), or proposing new energy-based algorithms to enhance robustness, realism and generalization to other soft bodies (Liu et al., 2017). Other works propose leveraging the parallel computational capabilities of modern GPUs (Tang et al., 2013; Zeller, 2005). These approaches achieve high realism at the expense of a great computational cost. Thus, PBS is not an appropriate solution when real-time performance is required or computational capacity is limited (e.g. in portable devices). On the other hand, for applications that prioritize performance, **LBS** (Linear Blend Skinning) is the standard approach on computer graphics for animation of 3D models. Each vertex of the object to animate is attached to a skeleton through a set of blend weights that are used to linearly combine joint transformations. In garment domain, outfits are attached to the skeleton driving body motion. This approach has also been widely studied (Kavan et al., 2008; Kavan et al., 2005; Le et al., 2012; Magnenat-thalmann et al., 1988; Wang et al., 2007; Wang et al., 2002). While it is possible to achieve real-time performance, cloth dynamics are highly non-linear, which results in a significant loss of realism when applied to garments.

Learning-Based. Due to the drawbacks found in the classical LBS approach, **PSD** (Pose Space Deformation) models appeared (Lewis et al., 2000). To avoid artifacts due to skinning, corrective deformations are applied to the mesh in rest pose. Additionally, PSD handles pose-dependant high frequency details of 3D objects. While hand-crafted PSD is possible, in practice, it is learnt from data. We find applications of this technique for body models (Allen et al., 2002; Anguelov et al., 2005; Loper et al., 2015a), where deformation bases are computed through linear decomposition of registered body scans. Similarly, in garment domain, Guan et al. (Guan et al., 2012) apply the same techniques for a few template garments on data obtained through simulation. Lähner et al. (Lahner et al., 2018) also propose linearly learnt PSD for garments, but conditioned on temporal features processed by an RNN to achieve a non-linear mapping. Later, Santesteban et al. (Santesteban et al., 2019) propose an explicit non-linear mapping for PSD through an MLP for a single template garment. The main drawback of these approaches is that PSD must be learnt for each template garment, which in turns requires new simulations to obtain the corresponding data. To address this issue, many researchers propose an extension of a human body model (SMPL (Loper et al., 2015a)), encoding garments as additional displacements and topology as subsets of vertices (Alldieck et al., 2018b; Alldieck et al., 2019; Bertiche et al., 2020; Bhatnagar et al., 2019; Patel et al., 2020). (Alldieck et al., 2018b; Alldieck et al., 2019) propose a single model for body and clothes, first as vertex displacements and later as texture displacement maps, to infer 3D shape from single RGB images. Similarly, (Bhatnagar et al., 2019) also learn a space for body deformations to encode outfits, plus an additional segmentation to separate body and clothes, also to infer 3D garments from RGB. (Jiang et al., 2020) propose 3D outfit retrieval from images and predicting the corresponding blend weights w.r.t. SMPL skeleton using, as labels, the weights of the nearest skin vertices. (Patel et al., 2020) encode a few different garment types as subsets of body vertices and propose a strategy to explicitly deal with high frequency pose-dependant cloth details for different

body shapes and garment styles. (Bertiche et al., 2020) encode thousands of garments on top of the human body by masking its vertices. They learn a continuous space for garment types, on which later they condition, along with the pose, the vertex deformations. Using a body model to represent garments allows handling multiple types with a single model. Nonetheless, it is still limited to single garments, as it cannot work with multiple layers of cloth. For the same reason, they cannot handle complex garment details. This reduces their applicability in real scenarios. Our proposed methodology allows working with arbitrary topologies, number of layers and complex details. Additionally, output format is highly efficient and allows easy integration into graphics engines, increasing compatibility and applicability.

4.3 Predicting Animated 3D Models

Computer graphics 3D animated models are constructed using skinning and/or blend shapes. In the former, given a 3D mesh with N vertices as $\mathbf{T} \in \mathbb{R}^{N\times 3}$ and a skeleton with K joints as $\mathbf{J} \in \mathbb{R}^{K\times 3}$, each mesh vertex is attached to each joint with a blend weights matrix $\mathbf{W} \in \mathbb{R}^{N\times K}$. Then, animating the 3D mesh can be achieved by posing skeleton $\mathbf{J}$ through linear transformation matrices (rotation, scaling and translation). Vertex transformation matrices are obtained as a weighted average of joint transformations as described by blend weights. For realistic human and garment animation, only rotations are applied to the joints, and thus, an axis-angle representation is used for pose as $\theta \in \mathbb{R}^{K\times 3}$. For the latter, given $\mathbf{T}$ as defined above, a blend shapes matrix as $\mathbf{D} \in \mathbb{R}^{M\times N\times 3}$ encodes M different deformations (shapes) $D_i \in \mathbb{R}^{N\times 3}$ for $\mathbf{T}$. To animate the mesh, M shape keys are required. These keys are used to linearly combine blend shapes to obtain a final deformation for $\mathbf{T}$. Temporal evolution of shape keys animates the mesh. More complex 3D models use a combination of both techniques. First, $\mathbf{T}$ is linearly deformed through blend shapes and later posed along skeleton $\mathbf{J}$ according to blend weights. Whenever shape keys are defined as a function of skeleton pose, we have Pose Space Deformations driven by *pose* keys. More formally, in human and garment animation domain:

$$\mathbf{V}_\theta = W(\mathbf{T} + \sum_i^M f(\theta)_i D_i, \mathbf{J}, \theta, \mathbf{W}) \tag{4.1}$$

Where $W(\cdot)$ is the skinning function that poses mesh vertices as described by $\mathbf{J}$ and θ, $\mathbf{V}_\theta$ is the posed vertices, $f(\cdot)$ is a function that maps pose θ to M pose keys and D_i are the shapes within blend shapes matrix $\mathbf{D}$. These techniques are the standard for 3D animation. All current graphics engines are compatible with these methods.

An example for this is SMPL (Loper et al., 2015a) (human body model). SMPL consists of a template mesh with vertices $\mathbf{T} \in \mathbb{R}^{6890\times 3}$, a skeleton $\mathbf{J} \in \mathbb{R}^{24\times 3}$, a blend weights matrix $\mathbf{W} \in \mathbb{R}^{6890\times 24}$ and two blend shapes matrices, one to represent different body shapes, $\mathbf{D}_{\text{shape}} \in \mathbb{R}^{10\times 6890\times 3}$, and another for Pose Space Deformations, $\mathbf{D}_{\text{PSD}} \in \mathbb{R}^{207\times 6890\times 3}$. Body shape is defined by shape keys $\beta \in \mathbb{R}^{10}$ and Pose Space Deformations by pose keys as flattened rotation matrices (removing global orientation) $R \in \mathbb{R}^{207}$. Because of its formulation SMPL is compatible with current graphics engines. Through this chapter, we use SMPL as support body model for animating outfits.

In this work we present a novel approach for garment animation. While recent works are already leveraging skinning blend weights w.r.t. body skeleton to drive

garment motion, authors usually rely on complex formulations for Pose Space Deformations, hindering their compatibility with graphics engines and reducing significantly their applicability in real scenarios. We propose learning a mapping from template outfit (canonical pose) meshes to their corresponding blend weights and blend shapes matrices through deep learning. That is, learning a neural network $\mathcal{M}$ as:

$$\mathcal{M} : \{\mathbf{T}, \mathbf{F}\} \rightarrow \{\mathbf{W}, \mathbf{D}_{\text{PSD}}\}, \tag{4.2}$$

Where $\mathbf{T}$ are outfit template vertices and $\mathbf{F}$ is mesh faces, $\mathbf{W}$ and $\mathbf{D}_{\text{PSD}}$ are the blend weights and blend shapes matrices as defined above. Note that in deployment, a template outfit is processed by the network only once into its standard animated 3D model format. Once blend weights and blend matrices are obtained, the outfit is used as any other 3D animated model. This makes predictions automatically compatible with all graphics engines, and furthermore, due to the exhaustive optimization of rendering pipelines for such models, it is an extremely computationally efficient representation. This further extends its applicability to portable devices and low-computing environments. It represents an advantage against other related works that predict vertex locations directly with neural networks (and often through large, complex models). Such approaches require major engineering efforts to adapt to real applications. Furthermore, due to memory footprint and computational cost of neural networks, these solutions might be impossible to use in low-computing devices. Finally, we also show how this approach allows generalization to unseen template outfits without retraining, which greatly enhances scalability.

4.4 Methodology

Given PBS data for outfits on top of human bodies (SMPL) in different action sequences, we define samples $\mathcal{S} = \{X, Y\}$ as $X = \{\mathbf{T}, \mathbf{F}, \theta, \beta, g\}$ and $Y = \{\mathbf{V}_{\text{PBS}}\}$, where $\mathbf{T}$ is the template outfit vertices (canonical pose), $\mathbf{F}$ is outfit mesh faces, θ is body skeleton pose, β is body shape parameters, g is body gender and $\mathbf{V}_{\text{PBS}}$ is the outfit vertex locations in the simulated data. Our goal is to train $\mathcal{M}$ as defined in Eq. 4.2 such that $\mathbf{W}$ and $\mathbf{D}_{\text{PSD}}$ yield $\mathbf{V}_{\text{PBS}}$ after applying Eq. 4.1 (Note that for SMPL, skeleton is a function of shape β and gender).

4.4.1 PBS Data and Physical Consistency

The mapping from pose-space to outfit-space is a multi-valued function. Different simulators, initial conditions, action speeds, timesteps and integrators, among other factors, will generate different valid outfit vertex locations for the same body pose and shape and outfit. Training on PBS data falsely assumes that this mapping is single-valued. Samples with similar X but significantly different Y will hinder network performance during training and most likely converge to average vertex location under a supervised loss. Moreover, a final user does not know the *ground truth* and therefore cannot perceive the accuracy of the model, but the user can assess the physical consistency of the predictions (collision-free and cloth consistency). Because of this, while resorting to PBS data for supervision is helpful for training networks, minimizing Euclidean error w.r.t. *ground truth* does not guarantee physical consistency, and therefore, the applicability of the predictions in real life is limited. Recent works (Patel et al., 2020; Santesteban et al., 2019) propose post-processing to solve body penetrations. This partially defeats the purpose of using deep-learning

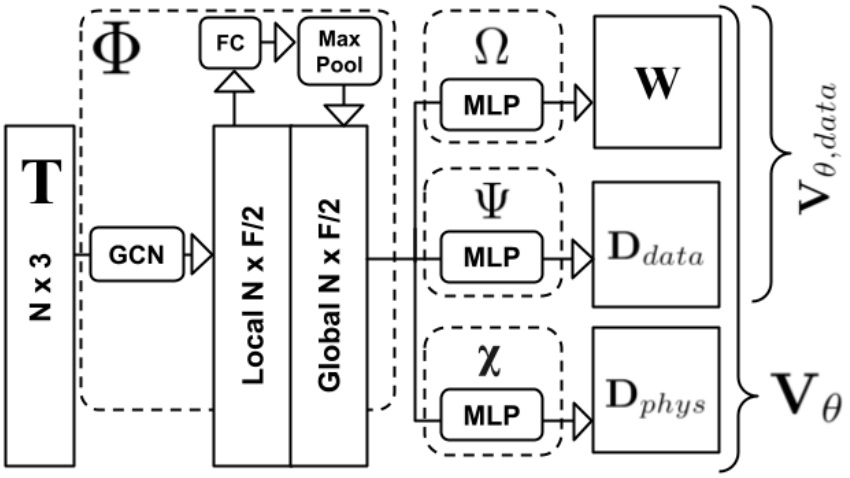


FIGURE 4.2: Model overview. The input of the model is a template outfit mesh (with no fixed topology, vertex order or connectivity). We apply graph convolutions to obtain vertex local descriptors. Then, local descriptors are processed by a fully-connected layer and aggregated through per-outfit max-pooling. This yields a global outfit descriptor that is concatenated with each vertex local descriptor. Final vertex descriptors are processed through different MLPs to obtain blend weights $\mathbf{W}$ and blend shapes matrices $\mathbf{D}_{data}$ and $\mathbf{D}_{phys}$. Blend shapes matrices are combined into $\mathbf{D}_{\text{PSD}}$, which is used as described in Eq. 4.1 to obtain final predictions. Pose keys for blend shapes matrix are obtained by passing θ through an MLP with 4 layers (not shown).

and further compromises method compatibility and performance. We propose combining supervised training with unsupervised physically based training to alleviate the need of post-processing.

Physical consistency is a crucial part of proper outfit animation. While other approaches develop complex solutions to better overfit to their PBS data and translate training wrinkles to predictions, their lack of physical constraints is detrimental for their usability in real applications. Physical consistency is not only limited to collisions, but also to edge distortion and surface quality. Abnormally stretched or compressed edges (w.r.t. to its template lengths) will create texture distortions (UV map edges do not change in length, but mesh edges do). Approaches that represent garments as a subset of body vertices cannot enforce an edge constraint, as their template is the body itself (original template is lost after registration against the body). Our proposal addresses garment animation independently of edition/resizing, and therefore, it is possible to leverage the original template outfits to enforce the edge constraint during learning.

4.4.2 Architecture

The chosen architecture needs to be able to: a) handle unstructured meshes (no fixed vertex order or connectivity) and b) compute non-linear deformations w.r.t. the pose θ (as cloth behaviour is highly non-linear). To do so, we define the following components: $\Phi : \mathbb{R}^{N\times 3} \rightarrow \mathbb{R}^{N\times F}$, $\Omega : \mathbb{R}^{N\times F} \rightarrow \mathbb{R}^{N\times K}$, $\Psi : \mathbb{R}^{N\times F} \rightarrow \mathbb{R}^{P\times N\times 3}$ and $\chi : \mathbb{R}^{N\times F} \rightarrow \mathbb{R}^{P\times N\times 3}$. Component Φ computes per-vertex high-level F-dimensional descriptors with local and global information from template outfit mesh (with $F = 512$), Ω computes per-vertex blend weights from vertex descriptors, Ψ generates a blend shapes matrix supervisedly (note that it is equivalent to per-vertex *blend shapes* matrices as $\mathbf{d} \in \mathbb{R}^{P\times 3}$) and χ generates a blend shapes matrix unsupervisedly that will yield physical consistency. Note that we define P pose keys for blend shapes

matrices, instead of the dimensionality of pose θ. We pass θ through an MLP to obtain a high-level embedding of pose as $\Theta \in \mathbb{R}^P$. The motivation for this is: a) controlling dimensionality P, and therefore, blend shapes matrix size and capacity and b) to allow modelling non-linearities from pose-space to vertex-space.

Fig. 4.2 shows an overview of the model. To learn Φ, we use 4 layers of graph convolutions applied to template mesh. This will yield a local descriptor, with no global information. Inspired by PointNet (Qi et al., 2017), we process each local descriptor through an additional fully-connected layer and aggregate all vertex descriptors through max-pooling (per outfit). We concatenate this global descriptor to each vertex local descriptor. Then, Ω, Ψ and χ are defined as MLPs, with 4 fully-connected layers each, applied to vertex descriptors (vertices are independent *samples*). The chosen architecture permits processing unstructured meshes with any vertex number, order and connectivity. This is a significant advantage against approaches that rely on the body model for garment representation (Bertiche et al., 2020; Patel et al., 2020), since it requires an expensive registration for each sample that introduces error in the data. Then, Ψ and χ both compute blend shapes matrices: $\mathbf{D}_{data}$ to minimize supervision loss and $\mathbf{D}_{phys}$ for physical consistency. Despite being independent branches, on deployment, both matrices are combined to obtain the final PSD matrix $\mathbf{D}_{\text{PSD}} = \mathbf{D}_{data} + \mathbf{D}_{phys}$, thus keeping the aforementioned compatibility with graphics engines. Finally, the MLP used to obtain the high-level pose embedding Θ consists on 4 fully-connected layers. The output of the model during training is $\mathbf{V}_{\theta,data}$ for $\mathbf{D}_{data}$ and $\mathbf{V}_\theta$ for $\mathbf{D}_{\text{PSD}}$.

4.4.3 Training

Our model combines both supervised and unsupervised training. The supervised part of the model corresponds to Φ, Ω and Ψ. The goal of this *submodel* is to minimize Euclidean error w.r.t. PBS data. Thus, for its training, we apply an standard L2 loss on predicted vertex locations:

$$\mathcal{L}_{data} = \sum \|\mathbf{V}_{\theta,data} - \mathbf{V}_{\text{PBS}}\|^2, \tag{4.3}$$

Then, the unsupervised part of the model corresponds only to χ. We define unsupervised losses to satisfy prior distributions based on physical constraints. First, to ensure cloth consistency of predictions, inspired by *mass-spring* model (most widely used PBS model for cloth), we define a cloth loss term as:

$$\mathcal{L}_{cloth} = \mathcal{L}_E + \lambda_B \mathcal{L}_B = \sum_{e \in E} \|e - e_{\mathbf{T}}\|^2 + \lambda_B \Delta(\mathbf{n})^2, \tag{4.4}$$

where $\mathcal{L}_E$ is the edge term and $\mathcal{L}_B$ is the bending term. Then, E is the set of edges of the given outfit mesh, e is the predicted edge length and $e_{\mathbf{T}}$ is the edge length on the template outfit $\mathbf{T}$. Then, $\Delta(\cdot)$ is the Laplace-Beltrami operator applied to vertex normals $\mathbf{n}$ of the predicted outfit and λ_B balances both losses. $\mathcal{L}_E$ enforces the output meshes to have the same edge lengths as the input template outfit, while $\mathcal{L}_B$ helps yielding locally smooth surfaces, as it penalizes differences on neighbouring vertex normals. To avoid excessive flattening, we choose $\lambda_B = 0.0005$. Then, to handle collisions against the body, we define a loss as:

$$\mathcal{L}_{collision} = \sum_{(i,j) \in A} min(\mathbf{d}_{j,i} \cdot \mathbf{n}_j - \epsilon, 0)^2, \tag{4.5}$$

where A is the set of correspondences (i, j) between predicted outfit and body through nearest neighbour, $\mathbf{d}_{j,i}$ is the vector going from the j-th vertex of the body to the i-th vertex of the outfit, $\mathbf{n}_j$ is the j-th vertex normal of the body and ϵ is a small positive threshold to increase robustness. This loss is a simplified formulation that assumes cloth is close to the skin, and penalizes outfit vertices placed inside the skin. In our experiments, we choose $\epsilon = 5$mm. Thus, the unsupervised loss is defined as:

$$\mathcal{L}_{phys} = \mathcal{L}_{cloth} + \lambda_{collision}\mathcal{L}_{collision} \tag{4.6}$$

where $\lambda_{collision}$ is the balancing weight for the collision term (around 2-10 in our experiments). Note how both terms $\mathcal{L}_{cloth}$ and $\mathcal{L}_{collision}$ are defined as priors (based only on X, not on Y). We define an additional loss term as an L2 regularization on deformations due to χ with a balancing weight $\lambda = 1e-2$. This leads χ to use deformations as small as possible to solve physical constraints. While the whole model is differentiable and could be trained end-to-end, we back-propagate $\mathcal{L}_{phys}$ only through χ. The motivation for this is:

- **Independent Tasks.** We empirically observed how supervised and unsupervised terms fight each other, compromising one or both tasks. Thus, by training different parts of the model independently, we do not need to find a balance between low Euclidean error and physical consistency. This allows the supervised submodel to learn the main deformations leveraging PBS data and the unsupervised branch to enforce physical consistency without their gradients hindering each other.
- **Unsupervised Training.** Since $\mathcal{L}_{phys}$ does not rely on Y, it is possible to train χ with new samples where θ is replaced in X by any other sample pose. This increases the amount of available data to train, enhancing generalization of physical consistency.

In practice, it is not helpful to train χ until the supervised training has converged.

4.5 Experiments

From the public datasets on garments, only CLOTH3D (Bertiche et al., 2020) contains enough outfit variability to implement this approach and achieve proper generalization. CLOTH3D was introduced in Chapter 3 as the largest by a great margin, specially in garment variability. It contains $\sim$ 7.5k sequences, each with a different template outfit in rest pose plus up to 300 frames. The outfits are simulated on top of an animated 3D human (SMPL), each with a different body shape. Likewise, we use SMPL skeleton in Eq. 4.1, so it drives the motion of the outfit, and its body mesh in Eq. 4.5. For the ablation study, we subsample 50k training *frames* and 5k test *frames* from CLOTH3D in a stratified manner w.r.t. sequences without outfit overlapping between both sets. Each model is trained for 10 epochs. We additionally present proof-of-concept computer vision applications as well as a performance analysis in the supplementary material.

4.5.1 Ablation study

We first evaluate the supervised part of the model (Φ, Ω and Ψ) by using the average vertex Euclidean error per outfit. In Tab. 4.1 we show the results to justify the design of the network. First, we propose a baseline model. In this baseline, global descriptor is not computed and Ψ predicts vertex deformations instead of blend shapes matrices by concatenating pose to vertex descriptors. The following models

	Euclidean error (mm)
Baseline	29.98
+Global	28.04
+GlobalLite	28.59
+Global+GCN	28.76
+Global with MLP	28.43
DeePSD	**25.13**
-without pose embedding	30.93

TABLE 4.1: Architecture ablation study. First, as a baseline, we train Ω and Ψ to predict vertex deformations instead of blend shapes matrices. Subsequent rows are baselines extensions (deformation prediction) with a global descriptor. *DeePSD* row corresponds to the architecture shown in Fig. 4.2. As it can be seen, predicting blend shapes matrices is the best performing approach.

	Euclidean error (mm)
DeePSD	25.13
+ SMPL shape/gender	25.15
+ Fabric	24.76
+ Tightness + Fabric	**24.66**
+ SMPL + Tightness + Fabric	25.01

TABLE 4.2: Conditioning to metadata available in CLOTH3D (Bertiche et al., 2020) for each sample. We concatenate metadata to each vertex descriptor: SMPL shape and gender, per-garment fabric and per-outfit tightness. As shown, body metadata hinders performance, while outfit metadata enhances it.

	Error	Edge	Bend	Collision
No phys.	24.66	1.27	0.031	11.59%
Phys.	33.75	1.13	0.029	1.29%
+poses	34.45	1.12	0.029	1.02%

TABLE 4.3: Unsupervised training. We measure cloth quality with average edge elongation/compression and bending angle between neighbouring vertex normals. For body collision, we show the ratio of vertices placed within the body.

	Euclidean error (mm)
Tshirt	25.77
Top	17.33
Trousers	14.50
Jumpsuit	17.23
Skirt	41.15
Dress	35.94
Total	23.95

TABLE 4.4: Final quantitative results per garment. Note how *tighter* garment types have a significantly lower error than others.

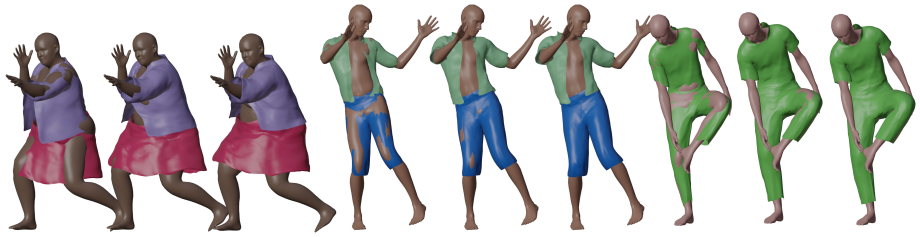


FIGURE 4.3: Qualitative results obtained by enforcing physical consistency. For each sample we show the results of each experiment in Tab. 4.3 in the same order from left to right.

are modifications of the baseline (predict deformations). The second row shows the result obtained by using global descriptors. It improves the accuracy of the predictions. The third row corresponds to a model with a lower descriptor dimensionality ($F = 128$), and we observe an slight increase in error. In the next experiment, we implement Ω and Ψ as graph convolutions instead of fully-connected layers. This worsens results at the cost of extra computational cost, thus we discard the use of graph convolutions after Φ. Note that this behaviour is expected, as global descriptor is broadcasted through vertices, and therefore, convolutions perform redundant information passes that hinder the learning. The next row corresponds to a model where global descriptor is obtained by replacing the single fully-connected layer in Φ with a MLP. Performance does not improve. *DeePSD* row corresponds to the architecture shown in Fig. 4.2. As one can observe, predicting blend shapes matrices instead of vertex deformations not only increases model compatibility with graphics engines, but it also improves performance. The final row corresponds to the same architecture as DeePSD, but using pose θ as pose keys instead of a high-level pose embedding. We see that predictions are less accurate, thus pose embedding Θ is beneficial.

We consider the effect of including additional metadata present in CLOTH3D. That is, SMPL body shape and gender, garment-wise fabric labels and outfit-wise tightness values. We combine these metadata by concatenating them to each vertex descriptor. Tab. 4.2 shows the quantitative results. The first row corresponds to the best model of Tab. 4.1. Each next row is named after the metadata used. As it can be observed, outfit metadata reduces Euclidean error while body metadata appears to be detrimental.

To evaluate the unsupervised model, we design suitable metrics for assessing cloth quality and physical constraints:

- **Edge Length.** Length difference between predicted and rest outfit edges, expressed in millimeters.
- **Bend Angle.** Cosine distance for pairs of neighbouring vertex normals.
- **Collision.** Ratio of collided vertices.

Edge metric summarizes cloth integrity. Cloth needs to compress or stretch to fit its environment in real life and PBS, thus, a zero-valued edge error might be impossible (even undesirable). Nonetheless, an abnormally high value suggests distorted predictions. Similarly, bend angle cannot be zero, otherwise we have a completely flat surface. Again, high values for this metric show poor cloth quality. Finally, for collisions, a zero-valued metric means physically consistent predictions. In practice, the training data contains invalid combinations of pose and shape (bodies with self-collisions), and therefore, a 0% of collided vertices is impossible. Tab. 4.3 shows the results for the ablation study of the physical consistency. First, we evaluate the predictions obtained with supervised loss only (best model of Tab. 4.2). Second row

shows the results obtained with χ trained without pose augmentation. The third row shows the results after training each sample with randomly chosen poses. We can observe that while Euclidean error increases, physical related metrics improve, specially collision. The model is learning to predict outfits farther from *ground truth* PBS data, but with higher physical consistency. As explained in Sec. 4.4.1, physical consistency cannot be summarized in one or few quantitative metrics. Results must be evaluated qualitatively. Fig. 4.3 shows a qualitative comparison of these experiments. As it can be seen, without physical constraints, although predictions have lower error by a large margin, qualitatively they are much worse. Also, we see that training unsupervisedly with randomly sampled poses further improves generalization.

We report final supervised results after fine-tuning with all data on Tab. 4.4. We decompose the error per garment. Note that *T-shirt* includes open shirts as well. We observe a worse performance for skirts and dresses. We also find a high error on *T-shirt*, likely due to open shirts. This is an expected behaviour, since modelling garments statically through skinning assumes the cloth will follow the body motion. Loose garments show a much more complex dynamics, and thus, static approaches will fail to model such garments. Fig. 4.1 shows qualitative results. We can see how the model can generalize to unseen complex outfits without retraining. Additionally, while cloth-to-cloth interaction is not explicitly addressed, the model is able to deal with multiple layers of cloth. It shows it can also handle complex geometric details (chest flower). As stated, it maintains cloth consistency, thus no artifacts appear on texturing. Finally, due to the unsupervised blend weights learning, skirts are robust against skinning artifacts due to leg motion (see supplementary material for more details on blend weights).

4.5.2 Comparison with related works

CLOTH3D. We compare DeePSD with CLOTH3D baseline quantitatively in Tab. 4.5 and qualitatively in Fig. 4.4. As it can be seen, our method outperforms CLOTH3D baseline. On the one hand, CLOTH3D baseline shows noisy boundaries and even broken suspenders. Furthermore, we observe the body geometry is present in the CLOTH3D reconstructed garment due to the use of SMPL body for garment representation. On the other hand, since DeePSD uses the original templates, boundaries are smooth and there is no bias to body geometry. Additionally, in spite of not dealing directly with cloth-to-cloth collisions, it appears that DeePSD is more robust in this aspect.

	Euclidean error (mm)
CLOTH3D (Bertiche et al., 2020)	29.0
DeePSD	**23.78**

TABLE 4.5: Comparison against CLOTH3D baseline. As CLOTH3D (Bertiche et al., 2020), we report the error garment-wise.

TailorNet. A fair quantitative comparison against the work of (Patel et al., 2020) is not possible. On one hand, TailorNet original simulations are not public, only the registered version against SMPL body. This means: a) original templates are lost and recovering them for each shape-style pair is unfeasible and b) their dataset has a fixed vertex order and connectivity (SMPL body). Since our main contribution is the generalization to unstructured meshes, comparing our methodology using a dataset

FIGURE 4.4: Qualitative comparison against CLOTH3D (Bertiche et al., 2020) baseline. Upper row: CLOTH3D. Lower row: DeePSD.

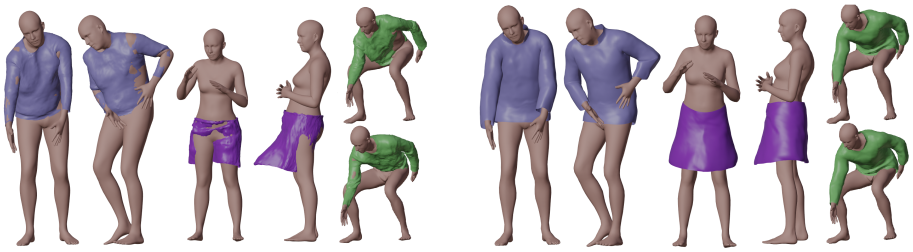


FIGURE 4.5: Comparison with TailorNet. Left: TailorNet. Right: DeePSD. TailorNet heavily relies in post-processing for valid predictions and generates noisy surfaces. The third sample (green T-shirt) shows two consecutive frames, note how TailorNet cannot guarantee temporal consistency.

with fixed vertex order against a methodology designed specifically for these data cannot be done fairly. On the the other hand, TailorNet is an ensemble of around 20 MLPs per each garment and gender which makes adapting it to CLOTH3D unfeasible, due to a much higher garment style variability. Thus, in Fig. 4.5, we compare TailorNet (left) and DeePSD (right) qualitatively. For fairness, since our approach uses no post-processing, we remove TailorNet post-processing. We gather similar garments and body shapes in TailorNet data and CLOTH3D and compute the same sequences using both models. As it can be seen, TailorNet is highly dependant on its post-processing due to a high amount of collided vertices. For the green T-shirt, samples correspond to consecutive frames. TailorNet cannot keep temporal consistency. DeePSD does not suffer from such effect. Similar to CLOTH3D baseline, we observe how body geometry is present on TailorNet predictions (leftmost sample chest) due to the use of SMPL to represent garments.

TailorNet succeeds in generating wrinkles in their predictions by overfitting an ensemble of MLPs per each garment type and gender. As stated by its authors: *"Our key simplifying assumption is that two garments on two different people will deform similarly..."*. Nonetheless, this has drawbacks. On one hand, as we have seen, it strongly compromises physical consistency, and thus, relies on post-processing. This increases sample generation times by 150-300ms. Note that applying a post-processing eliminates differentiability. Another drawback is the complexity of their model. Their ensemble of MLPs takes around 2GB per garment and gender. All of this hurts its applicability, compatibility and performance (and then, portability). On the contrary, DeePSD is a single small-sized model (4.4MB) that allows animating

any outfit (not only individual garments as body homotopies) without retraining. Predictions are generated as highly computationally efficient models (blend weights and blend shapes) compatible with any graphics engine. We obtain running times of 3-6ms for individual samples and around 0.1ms for batched samples (depending on vertex count). Furthermore, through physically based unsupervised learning, we alleviate the need of post-processing, thus maintaining differentiability and the aforementioned computational performance.

4.6 Conclusions and Future Work

We presented a novel approach for garment animation. Breaking the trend of previous approaches that try to predict vertex deformations through deep learning, we proposed learning a mapping from outfit space to animated 3D model space. We showed how this allows generalization to unseen outfits as well as compatibility with graphics engines. We observed how recent works need to leverage the body model for garment representation to allow edition/resizing along with animation, leading to overly complex models with scalability, compatibility and applicability issues. We addressed these issues by identifying garment animation as an independent task. We prioritized physical consistency in our predictions, relieving the need of post-processing. In summary, we developed an efficient approach applicable in real-scenarios as it is, even portable devices, that allows a more intuitive workflow for CGI artists that does not require expert knowledge in deep learning. Compare to the CLOTH3D baseline presented in Chpater 3, the garment quality has significantly increase. Mainly thanks to the possibility of using the original template garments. Lack of generative capacity, while it may look like a drawback, is actually a good thing. It is much easier for an end user to edit outfits in rest pose for later animation than trying to find a latent code that better approximates their desired design.

We observed limitations in our approach. First, loose garments, such as skirts and dresses, cannot be properly modelled with static approaches. To this end, we set as future work adapting our methodology to work with the temporal dimension. To keep its compatibility, pose keys should be computed with a temporal neural network while the training enforces dynamic learning (whether it is from data or unsupervisedly through physical consistency). We also observed how recent works grow on complexity to model fine geometric details (wrinkles). We believe a suitable approach to deal with garment wrinkles is through normal map generation because: a) it allows using lower vertex counts without compromising details, b) it is directly compatible with all graphics engines and c) it is more robust to collisions, since graphics engines compute face visibility on base geometry. Current works on this domain appear to be promising (Lahner et al., 2018; Zhang et al., 2020). We set this as future work. Nevertheless, this solution does not address the accurate 3D modelling of the cloth. Training a single model on thousands of garments with diverse wrinkle distribution converges to average predictions that lack high frequency details. This may also be in part due to supervised learning being sub-optimal for cloth. Next, Chapter 5 will focus on learning realistic cloth deformations for single outfits.

Chapter 5

PBNS: Physically Based Neural Simulation for Unsupervised Garment Pose Space Deformation

FIGURE 5.1: We present a neural simulator for outfits. Our methodology yields skinned models with Pose Space Deformations through an implicit Physically Based Simulation using deep learning framework. This figure shows different neurally simulated outfits on different unseen body poses. Our results do not need collision-solving post-processing.

5.1 Introduction

Animation of draped humans has been widely explored by the computer graphics community because of its wide range of potential applications: videogame, film industry, and nowadays, also in virtual and augmented reality VR/AR. We can split the different animation approaches based on their goal: performance or realism. On one hand, Physically Based Simulation (PBS) (Baraff et al., 1998; Liu et al., 2017; Provot, 1997; Provot et al., 1995; Tang et al., 2013; Vassilev et al., 2001; Zeller, 2005) strategies discretize the space and time to apply basic physics laws. The realism obtained is closely related to how fine-grained is the discretization. On the other hand, Linear Blend Skinning (LBS) (Kavan et al., 2008; Kavan et al., 2005; Le et al., 2012; Magnenat-thalmann et al., 1988; Wang et al., 2007; Wang et al., 2002) and Pose Space Deformation (PSD) (Allen et al., 2002; Anguelov et al., 2005; Lewis et al., 2000; Loper et al., 2015a) techniques require a significantly lower amount of computational resources but compromise the realism of the animation. These strategies are suitable for low-computing environments or applications that demand real-time performance (portable devices and videogames). We close the gap between PBS and

PSD by proposing an unsupervised approach to learn realistic PSD, and thus, maintaining their efficiency advantage against PBS.

Due to its recent success in complex 3D tasks (Arsalan Soltani et al., 2017; Han et al., 2017; Madadi et al., 2020; Omran et al., 2018; Qi et al., 2017; Richardson et al., 2016; Socher et al., 2012), we find deep learning as a promising approach to the garment animation problem. The research community has shown an increasing interest on draped 3D human animation through deep learning during the past few years (Alldieck et al., 2018b; Alldieck et al., 2019; Bertiche et al., 2020; Bhatnagar et al., 2019; Guan et al., 2012; Lahner et al., 2018; Patel et al., 2020; Santesteban et al., 2019). Commonly, authors propose learning non-linear PSD models from big volumes of PBS data. Hence, these approaches also demand high computational resources to run the simulations. Another possibility for data gathering is through the use of 4D scans. While this solution allows capturing real data, it is necessary to build expensive and constrained setups. Furthermore, data obtained through scans need to be post-processed to be usable. Supervised deep learning based approaches not only depend on expensive data, but are also bounded by it. Plus, supervised training falsely assumes uniqueness of garment vertex locations, which, as we will show, hinders learning in practice.

In this chapter we propose learning PSD for rigged garments leveraging a deep learning framework and formulating the problem as an implicit PBS. By using PBS formulation, we force our models to predict consistent, low-energy configurations of the physical system that cloth and body represent. Doing this allows applying unsupervised training, removing the need of data gathering through expensive simulations or scans. Furthermore, we show that our proposed methodology can yield cloth-consistent PSD in a short amount of time (minutes). By eliminating the need of simulating hundreds or even thousands of sequences, we drastically reduce the time needed from garment design to model deployment. This increases the applicability and scalability of the methodology, broadening the scope of real life scenarios that will benefit from it. The final animated garments show cloth-consistency, pose-dependant wrinkles and temporal coherence for unseen pose sequences. Fig. 5.1 shows some qualitative samples obtained with the methodology described in this chapter. Our main contributions are:

- **Unsupervised PSD Learning.** By enforcing physical consistency during the training of the model, we eliminate the dependency of PBS or scan data. As a consequence, this methodology can be applied to an arbitrary number of garments, body shapes and poses without the computational cost of obtaining data for them.

- **Efficient Training, Deployment and Compatibility.** Related deep based approaches in the current literature propose complex formulations to obtain realistic results. This hinders the training process and posterior deployment. Our proposed methodology yields blend shapes for LBS models, which is the standard for 3D animation, and therefore, it is automatically compatible with all graphic engines and benefits from the exhaustive optimization for these models. This greatly increases the applicability of the methodology.

- **Physical Consistency.** Learning based related works are unable to predict collision-free garments, and thus, require collision-solving post-processing to be used in real applications (Patel et al., 2020; Vidaurre et al., 2020; Jiang et al., 2020; Gundogdu et al., 2019b; Guan et al., 2012; Santesteban et al., 2019). This disables real-time performance, increases the required engineering effort to

adapt such solutions and removes model differentiability, which hurts their applicability in research. Some of these works use instead collision-solving losses during training to alleviate the issue, but body interpenetration still appears. We show how this is related to supervised training. On the other hand, PBNS can generate collision-free predictions even under extreme unseen poses, effectively removing the need of post-processing. Physical consistency is not limited to collisions, but also to surface quality. Inspired by mass-spring models, we enforce edge and bending constraints in our predictions. This ensures outfits have no distorted edges —which would generate texturing artifacts— and smooth surfaces.

- **Cloth-to-cloth Interaction.** We are the first to propose a learning-based approach that is able to explicitly handle cloth-to-cloth interactions between different layers of cloth. Because of this, PBNS is the only current approach that can animate complete and complex outfits with multiple overlapping garments.

Contrary to the solution presented in Chapter 4, this methodology is outfit-specific. Nonetheless, the quality of the results is noticeably improved, as well as its robustness against body collisions or artifacts.

5.2 State-of-the-art

In the computer graphics community, the garment animation problem has been tackled for decades. Although deep learning has shown significant progress during recent years, one of its main drawbacks in the case of garment animation is the scarcity of available data and the data-hungry nature of deep-based approaches.

5.2.1 Computer Graphics

PBS (Physically Based Simulation) permits obtaining highly realistic cloth dynamics, usually relying on the *spring-mass* model. The literature on this regard is extensive and mainly addresses the efficiency and robustness of the methodology. This is done through simplifications and specialization on constrained scenarios (Baraff et al., 1998; Provot, 1997; Provot et al., 1995; Vassilev et al., 2001). As another option, authors propose energy-based optimization approaches for an increase in stability and generalization to additional soft-bodies (Liu et al., 2017). Some works describe technical improvements to leverage the extra computational power that GPU parallelization yields (Tang et al., 2013; Zeller, 2005; Tang et al., 2018). Nonetheless, in spite of the increase on efficiency contributed by other works, achieving a high level of realism comes at a great computational expense. When such resources are not available and real-time performance is a must, PBS cannot be applied. To overcome this, **LBS** (Linear Blend Skinning) is used. LBS –and other skinning techniques– is the current standard for real-time 3D animation in computer graphics. Objects motion is driven by an skeleton defined as a set of joints. Vertices of the mesh that represent the 3D object are attached to the joints by a set of blend weights. The transformation (rotation, translation and scaling) of each vertex is the weighted sum of the transformations of the joints with the aforementioned blend weights. Commonly, garments are attached to the same skeleton that controls the 3D body. We can also find an extensive research regarding LBS (Kavan et al., 2008; Kavan et al., 2005; Le et al., 2012; Magnenat-thalmann et al., 1988; Wang et al., 2007; Wang et al.,

2002). This approach allows real-time applications, even in low-computing environments, by sacrificing realism, specially on garment domain. Currently, we can find **hybrid** strategies in the industry. Tight parts of the outfits (e.g., t-shirt and trousers) are attached to the body skeleton while other apparel (e.g., capes and long coats) are simulated. This approach is widely used in the videogame industry, as realism is enhanced without an excessive increase on computational requirements.

5.2.2 Learning-Based Approaches

LBS models achieve real-time performance, nonetheless, linear transformations are usually not enough to capture the motion of soft-tissue objects such as cloth. Furthermore, LBS might suffer of skinning-related artifacts. **PSD** (Pose Space Deformation) aims to address these drawbacks by applying corrective deformations to LBS models before skinning (Lewis et al., 2000). This helps reducing artifacts and also allows representation of high-frequency details that depend on the pose of the object. Hand-crafted PSD is too labor intensive for complex models (such as the human body) and it is usually learnt from data. Authors have shown that PSD approaches are able to model the human body (Allen et al., 2002; Anguelov et al., 2005; Loper et al., 2015a). Deformations basis are obtained by linear decomposition of hundreds or thousands of 3D body scans. Following this fashion, for garments, (Guan et al., 2012) propose applying the aforementioned methodology for synthetic garment data gathered through PBS. Later, (Lahner et al., 2018) extend the idea by computing the aforementioned linear decomposition against temporal feature arrays processed by a Recurrent Neural Network (RNN), achieving non-linearity w.r.t. the pose. (Santesteban et al., 2019) explicitly apply a non-linear mapping with a Multi-Layer Perceptron (MLP) for a single fixed garment. (Zurdo et al., 2012) show an example based methodology for non-skinned cloth for coarse simulation up-sampling. (Kim et al., 2013) compress tens of gigabytes of precomputed cloth simulations into an small motion graph that allows for fast garment animation. (Hahn et al., 2014) learn a set of low-dimensional subspace basis for fast garment simulation from training data. Similarly, (Xu et al., 2016) precompute pose-space basis to efficiently simulate soft-tissue dynamics on rigged characters. Then, (Chentanez et al., 2020) propose a convolutional network for triangular meshes for garment and soft-tissue animation and coarse simulation up-sampling. (Jin et al., 2020) present a pixel based solution in the UV space of garments, tackling the problem as a computer vision task. (Holden et al., 2019; Pfaff et al., 2020) show data-driven approaches for general neural physics. Also, (Geng et al., 2020) propose a physically based post-processing for garment predictions from a neural network to alleviate compression and stretching. (Bailey et al., 2018; Bailey et al., 2020) detail a deep-learning based methodology for a fast modelling of non-linear pose space deformations for body and face models. While these approaches achieve appealing results, when applied to garment animation, each new garment or outfit requires repeating the simulation and learning process, thus hindering scalability and applicability. Authors commonly address this drawback by leveraging existing body models (like SMPL(Loper et al., 2015a)). Garments are encoded on top of the human body as subsets of displaced vertices (Alldieck et al., 2018b; Alldieck et al., 2019; Bertiche et al., 2020; Bhatnagar et al., 2019; Patel et al., 2020). Following the idea of exploiting the human body model, (Patel et al., 2020) use subsets of body vertices as few different garments to later learn garment-specific models for high frequency cloth details. The authors of (Bertiche et al., 2020) perform a similar encoding for thousands of different garments. This allows learning a continuous space for garment topology. Later, (Bertiche et al., 2021d) propose a

graph convolutional network to predict blend weights and pose space deformations for any input mesh, allowing generalization to unseen garments regardless of their topology. Nonetheless, huge volumes of data are still needed to train these models. Furthermore, it has been proven that deep neural networks are biased to lower frequencies (Rahaman et al., 2019), and as noted by (Patel et al., 2020), this effect is more significant on cloth domain when training a single model to represent many different garment types. This means that in order to obtain high-resolution garment predictions, it is better to exhaustively simulate and train for individual garments. Additionally, most of the aforementioned works for cloth require post-processing on their predictions to be usable, hindering performance and model differentiability. Our proposed methodology allows skipping the simulation step and efficiently learn, in few minutes, PSD for a given LBS model of a garment or outfit. Furthermore, our predictions are physically consistent and do not require post-processing.

5.3 Neural Cloth Simulation

Classical computer graphics approaches resort to skinning and/or costly simulation, which compromise realism, performance or applicability. On the other hand, learning based approaches, non-deep and deep, propose a skinning followed by a data-driven method to compute Pose Space Deformations. Since simulated data is still necessary, a significant computational investment is required for each new garment, body shape or fabric. We propose learning garment-specific (or outfit-specific) PSD unsupervisedly by enforcing physical laws, addressing the main drawbacks of previous works in terms of data requirements.

5.3.1 PBS Data and Physical Consistency

As aforementioned, the current trend on this domain is supervised learning from PBS data. We will show how this is suboptimal for learning valid garment deformations. The mapping from pose-space to outfit-space is a multi-valued function. Different simulators, initial conditions, action speeds, timesteps and integrators, among other factors, will generate different valid outfit vertex locations for the same body pose and shape and outfit. Training or evaluating on PBS data falsely assumes that this mapping is single-valued. Samples with similar θ but significantly different vertex locations will hinder network performance during training and most likely converge to average vertex locations under a supervised loss. Moreover, a final user does not know the *ground truth* and therefore cannot perceive the accuracy of the model w.r.t. PBS data, but the user can assess the physical consistency of the predictions (collision-free and cloth consistency). Minimizing Euclidean error w.r.t. *ground truth* does not guarantee physical consistency, and therefore, the applicability of the obtained predictions in real life is limited. Recent works require post-processing to solve body penetrations (Patel et al., 2020; Santesteban et al., 2019). This partially defeats the purpose of using deep-learning, removes differentiability and real-time performance. Thus, standard PBS is always preferred against previous deep-based approaches. We propose a fully unsupervised training as an implicit physically based simulation to remove the need of post-processing. Because of this, and the extreme efficiency of our formulation, ours is the first approach that can be applied for real-time scenarios in most devices.

5.3.2 Formulation

Our goal is to obtain cloth-consistent PSD for a given garment or outfit rigged to the skeleton of an LBS body model, in order to animate cloth and body at once. Following (Loper et al., 2015a), we denote the per-vertex formulation of skinning with PSD w.r.t. an articulated skeleton, represented as a set of joints $J \in \mathbb{R}^{K\times 3}$, for a given template garment or outfit in rest pose $\mathbf{T} \in \mathbb{R}^{N\times 3}$ as:

$$t'_i = \sum_k^K w_{k,i} G_k(\boldsymbol{\theta}, J)(t_i + dt_i(\boldsymbol{\theta})), \tag{5.1}$$

where $w_{k,i}$ is a single blend weight for vertex i and joint k, G_k is the linear transformation matrix corresponding to joint k, $\boldsymbol{\theta}$ is the skeleton pose in axis-angle representation, t_i is the i-th garment vertex in rest pose and dt_i is the pose space deformation corresponding to this vertex. We need to find a valid skinning as a blend weights matrix $\mathcal{W} \in \mathbb{R}^{N\times K}$ and a PSD as a mapping $f : \boldsymbol{\theta} \rightarrow \{d\mathbf{T} \mid \mathbf{T}_{\boldsymbol{\theta}} = \mathbf{T} + d\mathbf{T}\}$ such that the output unposed garment $\mathbf{T}_{\boldsymbol{\theta}} \in \mathbb{R}^{N\times 3}$ is properly aligned with the body (no collisions) and shows a realistic cloth-like behaviour after skinning. We propose using a neural network to approximate f. Note that our formulation is not dependant on the chosen body model, and the only requirement is to have a database of poses for the body.

Blend Weights. In the current literature we find authors that rely on the assumption that garments closely follow body motion (Santesteban et al., 2019; Bertiche et al., 2020; Patel et al., 2020). Results presented by these works prove it is a valid assumption that allows for a significant simplification of the problem. We rely on this to compute blend weights for our template garment. For each vertex $t_i \in \mathbf{T}$ we assign blend weights equal to those of the closest body vertex, with the body in rest pose (aligned with garment). Skirts break the assumption that cloth and skin are close to each other. For these kind of garments, we allow blend weights to be optimized along with network weights. We observed that doing this increases the model convergence speed. For other types of garments, we see no significant differences, except the computational overhead of optimizing blend weights along with the rest of the trainable parameters.

PSD. Skinning alone is not enough to properly model garments, as cloth behaviour is highly non-linear. For this reason, we formulate the model with PSD. Classical computer graphics approaches rely on linear decomposition from training samples to obtain a PSD matrix like $\mathbf{D} \in \mathbb{R}^{|\boldsymbol{\theta}|\times N\times 3}$, where $|\boldsymbol{\theta}|$ is the dimensionality of the pose array and N is the number of vertices of the 3D mesh to animate. We propose to first obtain a high level embedding of the pose array $\boldsymbol{\theta}$ through a neural network as $\mathbf{X} = f_{\mathbf{X}}(\boldsymbol{\theta})$ and use this embedding to obtain the final deformations with a PSD matrix as $\mathbf{D} \in \mathbb{R}^{|\mathbf{X}|\times N\times 3}$. This allows flexible non-linear mapping from $\boldsymbol{\theta}$ to dT and also to control matrix $\mathbf{D}$ size and capacity. Learning of both $f_{\mathbf{X}}$ and $\mathbf{D}$ is performed following a deep learning framework, where variables are optimized by minimizing a loss function with batches of different input pose samples.

5.3.3 Architecture

We design $f_{\mathbf{X}}$ as a Multi-Layer Perceptron (MLP). More specifically, it consists of 4 fully connected layers with a dimensionality of 32 and ReLU activation function. Then, to obtain the posed outfit $\mathbf{V}_{\boldsymbol{\theta}}$ for a given $\boldsymbol{\theta}$:

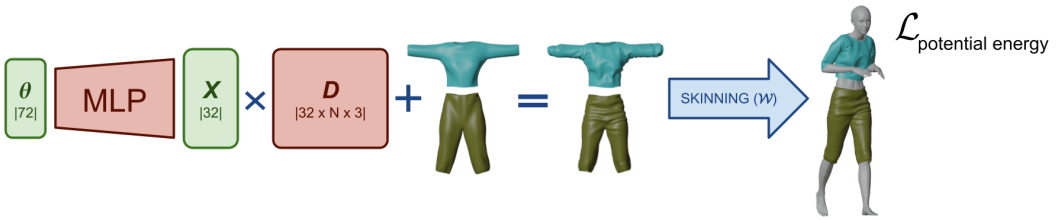


FIGURE 5.2: Methodology overview. Pose parameters are fed to a 4-layer Multi-Layer Perceptron with 32 dimensions in each layer. The output high-level pose embedding is multiplied with a PSD matrix $\mathbf{D}$ to obtain deformations for the garment/outfit. As standard, deformations are applied in rest pose garments and skinned along with the body (note that garment blend weights are assigned by proximity w.r.t. the body in rest pose). Finally, we train by optimizing the potential energy of the output physical system represented by the body and the cloth. The parts of the model in red correspond to the trainable parameters.

$$\mathbf{V}_{\boldsymbol{\theta}} = W(\mathbf{T} + f_{\mathbf{X}}(\boldsymbol{\theta}) \cdot \mathbf{D}, \boldsymbol{\theta}, \mathcal{W}) \tag{5.2}$$

Where $W(\cdot, \boldsymbol{\theta}, \mathcal{W})$ is the skinning function for pose $\boldsymbol{\theta}$ and blend weights $\mathcal{W}$, and the product $f_{\mathbf{X}}(\boldsymbol{\theta}) \cdot \mathbf{D}$ is computed as $\sum_i^{|\mathbf{X}|} f_{\mathbf{X}}(\boldsymbol{\theta})_i \mathbf{D_i}$. Under this formulation, we can approximate non-linear mappings from $\boldsymbol{\theta}$ to $\mathbf{V}_{\boldsymbol{\theta}}$ while keeping compatibility with current graphic engines. Fig. 5.2 shows an overview of the proposed methodology. The input of the model is the pose array $\boldsymbol{\theta}$, which is processed through the aforementioned MLP to yield a high-level pose embedding $\mathbf{X}$. This embedding $\mathbf{X}$ is multiplied with the PSD matrix $\mathbf{D}$ to obtain garment vertex deformations. Both, the MLP and the matrix $\mathbf{D}$ are learnt through training (and blend weights $\mathcal{W}$ are optimized from their initial proximity-based values for outfits with skirt). Finally, the deformed garment (or outfit) is skinned along the body according to $\boldsymbol{\theta}$ and blend weights $\mathcal{W}$. For the rest of the chapter, for clarity reasons, we consider the PSD matrix $\mathbf{D}$ to be part of the network. While it should be possible to apply this methodology without an MLP, we observe it presents important advantages. First, the size of matrix $\mathbf{D}$ is several orders of magnitude larger than the proposed MLP. Thus, controlling the size of this matrix allows for more efficient models. Without an MLP, the dimensionality of $\mathbf{D}$ is fixed to $\mathbb{R}^{|\theta| \times N \times 3}$ ($|\theta| = 72$ blend shapes for SMPL as body model), and each of these blend shapes would be associated to a single θ parameter. This is clearly sub-optimal as some of these blend shapes would be irrelevant. Also, blend shapes would be linearly tied to their corresponding pose parameters. This is also sub-optimal since the mapping from axis-angle space to Euclidean space is non-linear. Moreover, cloth deformations are likely to be non-linear w.r.t. body pose. With an MLP, the model learns more meaningful blend shapes that are combined non-linearly w.r.t. θ. Empirically, we also find the training to be faster and more stable with an MLP.

5.3.4 Training

Just as physical systems are implicitly *optimized* by acting forces $\mathbf{F} = -\nabla U$ (where U is the potential energy of the system), we train our neural network under a loss defined as a potential energy. This way, our model will learn to predict consistent, low-energy stable configurations. Note that during back-propagation, $\mathbf{F} = -\nabla U$ is explicitly computed. Thus, our methodology is implicitly simulating the network weights as a physical system for a given dataset of poses. We define this as neural simulation. We define our global loss (or potential energy) as:

$$\mathcal{L} = \mathcal{L}_{\text{cloth}} + \mathcal{L}_{\text{collision}} + \mathcal{L}_{\text{gravity}} + \mathcal{L}_{\text{pin}}, \tag{5.3}$$

where $\mathcal{L}_{\text{cloth}}$ corresponds to the elastic potential energy of the garment, and guides the model to predict cloth-consistent meshes. The term $\mathcal{L}_{\text{collision}}$ formulates body penetrations as a potential energy, thus its gradients will push cloth vertices to valid locations. Finally, $\mathcal{L}_{\text{gravity}}$ is the gravitational potential energy, which will minimize vertices height within the constraints set by the other loss terms. Additionally, we define $\mathcal{L}_{\text{pin}}$ to regularize deformations of chosen vertices (inspired by PBS).

Cloth consistency. The first term of our loss is related to the cloth consistency of the predictions. That is, we want the output meshes to fulfill certain properties we find in cloth. Classical computer graphics approaches usually rely on the mass-spring model to simulate cloth. We extend this idea to deep learning by designing a cloth loss term as:

$$\mathcal{L}_{\text{cloth}} = \lambda_e \mathcal{L}_{\text{edge}} + \lambda_b \mathcal{L}_{\text{bend}} = \lambda_e \left\| E - E_{\mathbf{T}} \right\|^2 + \lambda_b \Delta(\mathbf{N})^2, \tag{5.4}$$

where $E \in \mathbb{R}^{N_E}$ is the predicted edge lengths, $E_{\mathbf{T}} \in \mathbb{R}^{N_E}$ is the edge lengths on the rest garment $\mathbf{T}$ (with N_E as the number of edges in the mesh), $\mathbf{N} \in \mathbb{R}^{N_F \times 3}$ is the face normals (for N_F triangular faces), $\Delta(\cdot)$ is the Laplace-Beltrami operator and λ_e and λ_b are balancing factors. On the one hand, the term $\mathcal{L}_{\text{edge}}$ ensures that cloth is not excessively stretched or compressed. It is formulated as the potential elastic energy of the system, such that its gradients act as forces. On the other hand, $\mathcal{L}_{\text{bend}}$ enforces locally smooth surfaces by penalizing differences between neighbouring face normals (hinge-like forces). Note that the latter is computed taking into account face connectivity, not vertex.

Collisions. Next, the model needs to handle collisions with the body model. To do so, we design the following loss:

$$\mathcal{L}_{\text{collision}} = \lambda_c \sum_{(i,j) \in A} min(\mathbf{d}_{j,i} \cdot \mathbf{n}_j - \epsilon, 0)^2, \tag{5.5}$$

where A represents the set of correspondences (i, j) between predicted outfit and body, respectively, through nearest neighbour, $\mathbf{d}_{j,i}$ is the vector going from the j-th vertex of the body to the i-th vertex of the outfit, $\mathbf{n}_j$ is the j-th vertex normal of the body, ϵ is a small positive threshold to increase robustness and λ_c is a balancing weight ($\epsilon = 4mm$ and $\lambda_c = 25$ in our experiments). This loss is crucial to obtain valid predictions and its gradients will push outfit vertices outside the body. It is designed under the assumption that cloth closely follows the skin, which we can safely assume given that initial skinning blend weights are assigned by proximity. While this is a naive implementation of a collision loss, it works well in practice and similar L1 formulations have already been used in deep-learning based approaches (Tiwari et al., 2020; Jiang et al., 2020; Gundogdu et al., 2019b). We opted for a quadratic term to enhance generalization and stability, plus, it helps achieving a balance w.r.t. the other loss terms. Note that, as with PBS, invalid bodies (self-collided) might corrupt the results.

Cloth-to-cloth. To be able to model whole outfits, it is necessary to explicitly handle cloth-to-cloth interactions. To this end, we define a layer order for each garment of a given outfit, from inner to outer. Then, we iteratively apply Eq. 5.5 to each layer, computing correspondences A using body and previous layers. This will simulate repelling forces for both vertices $(i, j) \in A$ whenever correspondences connect two different cloth layers. To the best of our knowledge, we are the first to explicitly

tackle cloth-to-cloth interaction for learning based approaches.

Gravity. We include an additional term to enforce more realistic garment predictions. This term models the effect of the gravity. From classical mechanics, we know that potential gravitational energy is $U = m \cdot g \cdot h$, where m is the mass of the object, g is the gravity and h is the height of the object. Since m and g are constant, we can understand this loss as $\mathcal{L}_{\text{gravity}} = k\mathbf{V}_{\boldsymbol{\theta}z}$. In other words, we are minimizing the Z coordinate (vertical axis) of each vertex of the predicted garments.

Pinning. For some garments we want certain vertices not to move around. For example, lower body garments might fall down as training progresses due to the gravity loss. We want to restrict waist vertices deformation such that it remains attached to its original position. The concept of pinning appears in most cloth simulators. To this end, we implement an L2 regularization loss on the deformations dt of each vertex defined as pinned down. That is, $\mathcal{L}_{\text{pin}} = \lambda_{\text{pin}} \sum_i b_i dt_i^2$ where $b_i = 1$ if vertex i is *pinned*, else $b_i = 0$. Note that a hard constraint would most likely produce collisions against the body, so vertices need to be able to move slightly. Then, we include it in the loss as an extra term with a balancing weight $\lambda_{\text{pin}} = 10$.

By formulating both $\mathcal{L}_{\text{cloth}}$ and $\mathcal{L}_{\text{gravity}}$ as physical magnitudes, their corresponding loss balancing weights are directly related to the properties of the fabric we want to simulate: Young's modulus for the elasticity and its mass for gravity. This provides explainability to the approach. The rest of the losses, as with classical computer graphics PBS, are simplifications of the underlying physics. The only requirement for the potential energy is to be differentiable.

5.4 Experiments

In this section we will first describe the process to apply the methodology explained in Sec. 5.3. Then, we define the data and setup for the experimental part.

5.4.1 Body model

SMPL (Loper et al., 2015a) is the current standard in the literature for human analysis and garment animation. This model is an LBS with a linear PSD obtained through thousands of accurate 3D scans of different subjects. Its underlying skeleton is defined as a set of $K = 24$ joints. Public pose databases are available for this model (AMASS (Mahmood et al., 2019)). We then choose SMPL for the experimental part because both model and pose data are available to the public. Nonetheless, the methodology described in this chapter is compatible with any 3D model rigged to an skeleton. SMPL also allows generating different body shapes through blend shapes. During neural simulation, body shape is fixed (just as with PBS, where, in general, we do not want the body shape to change during simulation).

5.4.2 Template outfit

Once a body model is selected, a garment or outfit is designed for the body in rest pose, with an approximate resolution of 1cm in our experiments. We smooth templates as much as we can before neural simulation. This will ensure that high frequency details and deformations are indeed generated by the model from the pose. Then, initial blend weights for the cloth are obtained by proximity to the body.

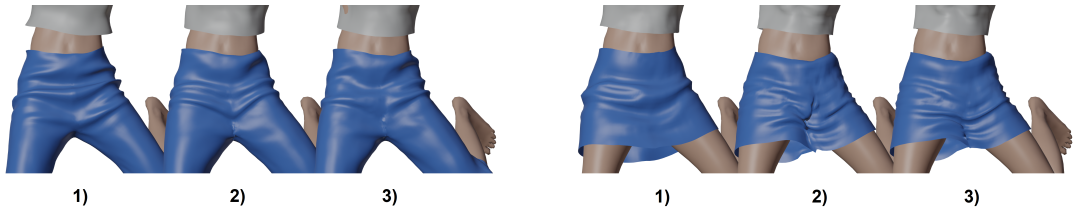


FIGURE 5.3: Architecture ablation study. We perform three experiments to assess the effect of the proposed MLP: 1) PBNS as described, 2) θ as input with no MLP and 3) rotation matrices $\mathbf{R}_\theta$ as input with no MLP. For experiment 2 we observe unrealistic V-shaped wrinkles around the hip. Also, for both models with no MLP, we observe artifacts where both legs merge. This effect is more evident on skirts. The merging point of left and right side of the skirt presents an important artifact for experiments with no MLP, despise being trained during hundreds of epochs (as opposed to 15 epochs for PBNS with MLP).

5.4.3 Pose database

Neural simulation requires a database of valid poses for the selected body model. We define, as valid poses, those that do not produce self-collisions when applied to the 3D body model. As with regular PBS, neurally simulating cloth over bodies with self-collisions will generate inconsistent repelling forces and might corrupt the results. For SMPL, we have $|\boldsymbol{\theta}| = 3K = 72$. We choose CMU MoCap pose sequences. This dataset contains 2667 pose sequences of different length, performed by different subjects. It totals around 4.3M individual poses. We split the database into train and test per subject, thus ensuring no subject or sequence is repeated in both sets, as 85/15%. Then, to ensure pose balance, we randomly sample $N = 3000$ poses from the training set, such that not any pair of poses have a distance $d < 0.5$, with $d = max(|\boldsymbol{\theta}_i - \boldsymbol{\theta}_j|)$. Thus, for any two poses, there is at least one parameter with a difference equal or bigger than 0.5 radians (we omit global orientation for this sampling). Later, we split the $N = 3000$ samples into training and validation set as 85/15% (2550 training poses and 450 validation poses).

5.5 Results

In this section we present the results obtained through experiments. First, a justification of the chosen architecture for the model. Then, a comparison against standard supervised learning. Later, we display and discuss qualitative results. Next, we show some interesting properties of neural simulation. Following, a comparison against the state-of-the-art. As it is standard in deep learning, all the presented results, quantitative or qualitative, correspond to unseen test sequences. Finally, we illustrate more possibilities for neural simulation: outfit resizing and custom avatar enhancement.

5.5.1 Multi-Layer Perceptron

In Sec. 5.3.3 we discussed the motivation for the MLP. We found it is possible to apply this methodology without an MLP by using pose θ to linearly combine the blend shapes within matrix $\mathbf{D}$ ($f_\mathbf{X}(\theta) = \theta$ in Eq. 5.1). We perform three different experiments to study the effects of an MLP: 1) PBNS as proposed in this chapter, 2) without MLP w.r.t. θ and 3) without MLP w.r.t. the rotation matrices $\mathbf{R}_\theta$ generated from θ (to alleviate non-linear relations between axis-angle space and Euclidean

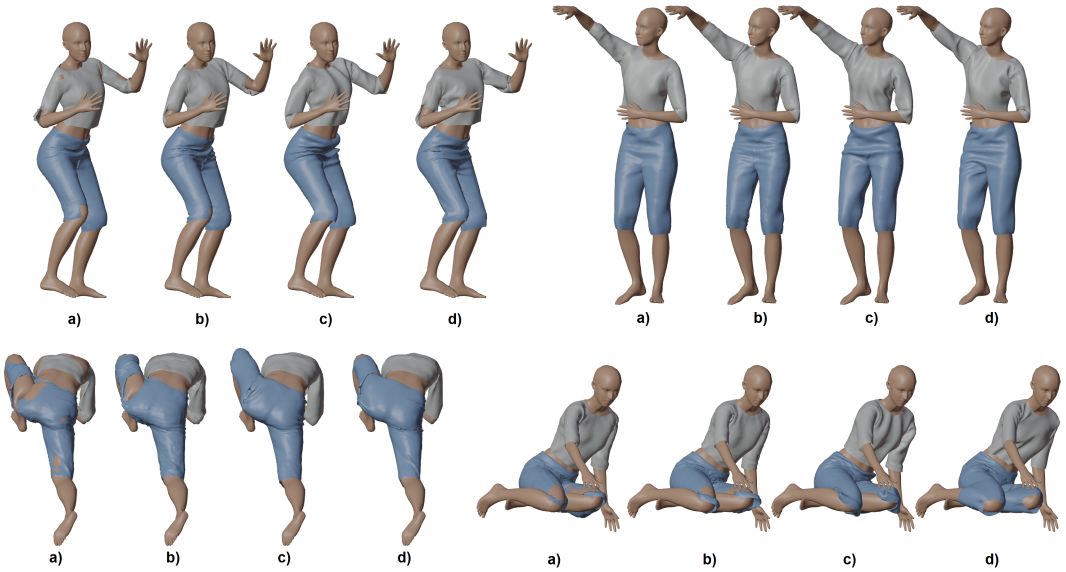


FIGURE 5.4: Supervision vs PBNS. We compare results obtained with PBNS against a purely supervised approach and a hybrid approach (L2 plus Eq. 5.3). The same architecture and the same $N = 3000$ poses (2550 for training and 450 for validation) are used in these experiments. We additionally compare against PBS data used for supervised training. We show: a) L2 only, b) hybrid, c) PBNS and d) PBS data. The supervised approach has the lowest Euclidean error w.r.t. PBS data, but also lowest number of wrinkles and highest number of collisions. The hybrid approach has more visible wrinkles and fewer collisions, but generalizes poorly to extreme poses. PBNS has the highest Euclidean error, but realistic pose-dependant wrinkles and shows no collisions, even under extreme poses. We also observe how PBS is prone to failures when body presents self-collisions (upper-left sample right elbow and lower-right sample legs). These failures might be transferred to predictions if trained supervisedly. PBNS is more robust to collisions than PBS.

space). Fig. 5.3 presents the results of these experiments. Each sample has a number that corresponds to one of the experiments explained. We see how experiment 2 presents unrealistic V-shaped wrinkles around the hip (left samples). Experiment number 3 does not show this. It means this is the result of linearly approximating a non-linear relation between axis-angle and Euclidean space. On top of this, we also observe artifacts on the region where both legs merge for experiments with no MLP. This effect is more evident on skirts (right samples). These artifacts are present even after hundreds of training epochs (note that experiment 1 is trained for just 15 epochs).

5.5.2 Supervised learning and quantitative evaluation

To compare our approach against supervised learning and provide of a quantitative metric, we compute PBS data for the $N = 3000$ poses described in Sec. 5.4.3. Once these data is obtained, we perform three different experiments (shown in Tab. 5.1). In the first row, we train the described architecture supervisedly with an standard L2 loss on the predictions w.r.t. PBS data. In the second row, we train the same model with L2 loss combined with Eq. 5.3. In the last row, we evaluate our unsupervised training results against the PBS data. We complement this table with a qualitative comparison shown in Fig. 5.4. From left to right: a) L2 only, b) hybrid, c) PBNS and d) PBS data. As can be seen, the supervised approach is able to minimize Euclidean

Method	Error (mm)	Edge (mm)	Collision	Time
L2	7.59	0.78	3.15%	$\sim$ 30h
Hybrid	8.21	0.74	1.08%	$\sim$ 30h
PBNS	15.52	0.66	0.45%	$\sim$ 15m

TABLE 5.1: Quantitative comparison of supervised baseline vs. PBNS. Each row represents an experiment. All experiments were performed with the same architecture. First row shows a purely supervised experiment, with an L2 loss w.r.t. PBS data. The second row combines L2 loss with Eq.5.3. Finally, the last row corresponds to PBNS as described in this chapter. For each experiment we report Euclidean error against PBS data, average edge compression/elongation w.r.t. edge rest lengths, the ratio of collided cloth vertices (within human body) and the time it takes to obtain a trained model. We observe how supervision compromises physical consistency (edge distortion and higher collision ratio). Since supervision requires PBS data, simulation time has to be taken into account. On the other hand, while Euclidean error is higher, PBNS can generate cloth-consistent and almost collision-free predictions in a very short amount of time.

error w.r.t. PBS data. Nonetheless, in order to do so, it compromises physical consistency. The supervised approach is prone to collisions, which makes predictions unusable in real applications. We also observe a minimal amount of wrinkles. The hybrid approach shows a lower number of collided vertices and slightly more visible wrinkles. Nonetheless, as can be seen, it generalizes poorly to extreme poses (lower row samples). Also, in the upper-left sample, we see a failure in the left elbow that is not present on PBS data. Thus, combining L2 loss with physical consistency has an unpredictable behaviour. Then, we see how PBNS can generate cloth-consistent and collision-free predictions, even under extreme poses. Finally, we also show PBS data in the figure. As can be seen, PBS might fail for poses that present body self-collisions (upper-left sample right elbow and lower-right sample legs). PBS failures might be transferred to predictions through supervision. Note how PBNS is more robust to failures than classic PBS. Human-like skinned models are prone to self-collisions, and thus, we consider this property a significant advantage over PBS. Finally, we also compare the amount of time devoted to obtain each model in the table. For the supervised and hybrid approaches, simulations are needed. New outfits and/or bodies will require new data. Thus, we include the time devoted to simulations in Tab. 5.1 In the time needed to obtain a single animated outfit through supervised approaches, PBNS can generate over a hundred of different models for different outfits. Overall, we have shown how unsupervised training is not only more efficient (no need to generate PBS data), but it also qualitatively outperforms supervised approaches. Furthermore, it shows higher robustness against simulation failures than traditional PBS. We have also shown how Euclidean error is misleading (lower does not mean better).

5.5.3 Qualitative

For a qualitative evaluation of the results, we refer first to Fig. 5.1. In this image we show a few samples with different pose, outfit and, one of them, different body. As it can be seen, our learnt PSD can generate appropriate wrinkles around bent joints in a realistic manner to fulfill the energy balance requirements imposed by our loss during training. Then, for a more in-depth analysis, we refer to Fig. 5.5. Here we show, on one hand, the template outfit of each sample. Templates are smoothed as much as possible. Later, for each template, the output for two extra unseen poses are

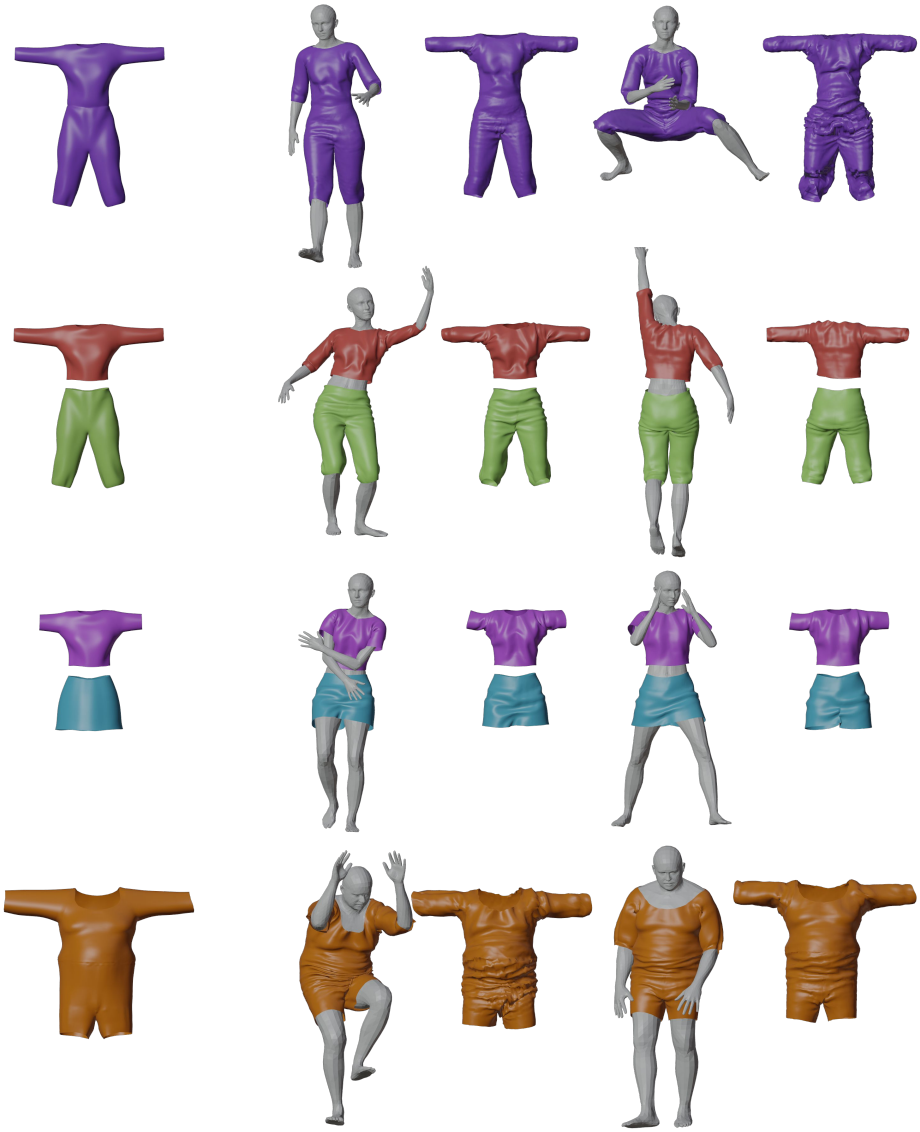


FIGURE 5.5: Qualitative results. Leftmost, we depict the template outfit of each row. Note how templates are smoothed as much as possible. Then, for each row we visualize the output for two unseen test poses. For each sample, we show the output dressed human (left) along with its corresponding deformed outfit in rest pose (right). As it can be seen, the learnt Pose Space Deformations generate folds and wrinkles to fulfill the energy balance requirement of the physical system. The last row depicts results of simulation with a different body shape (and its appropriately aligned template outfit) to show generalization to different bodies. All of the shown samples were obtained after just a few minutes of training.

FIGURE 5.6: Textured qualitative samples. This rendering gives an approximate idea to the final application-level looks that our methodology yields. These results correspond to neural simulations of 3 additional outfits on top of 3 different bodies, further showing the generalization capabilities of our methodology.

shown (different from Fig. 5.1). For each of these samples, the final rigged draped human is visualized (left) along with the unposed deformed template garment $\mathbf{T}_{\theta}$ (right). Templates show deformations to satisfy the energy constraints which would not be possible with skinning alone. From the first row, we can notice how big deformations are due to collisions for extreme poses. Also, while deformed template can look noisy, it looks realistic after skinning. On the second row we can see deformations on the back of the outfit. Nonetheless, as human bodies usually bend forward, wrinkles in the front are more evident. The third row shows samples with a skirt. Note how in the second sample, the skirt deformations need to correct rotations due to leg movements (deformed template has a discontinuity). For outfits with skirt, we allow optimization of blend weights, along with the rest of the network, to alleviate the correction required due to leg motion. In spite of this, the effect on the template is not fully mitigated. Finally, the last row shows results obtained with a different body shape. As aforementioned, the methodology presented in this chapter is compatible with any 3D model rigged to an skeleton. On Fig. 5.6 we rendered more qualitative results for different bodies and outfits. All of these visualizations correspond to models trained during just a few minutes without any post-processing.

5.5.4 Multiple Layers and Controllable Parameters

The proposed collision loss can be extended to deal with multiple layers of cloth (see Sec. 5.3.4). Fig. 5.7 contains the results obtained for some outfits that present cloth-to-cloth interaction. For each sample, from left to right: outfit in rest pose, posed outfit without outer layer, whole posed outfit and cross section. First, we see how templates in rest pose show collisions, against the body and cloth-to-cloth. The collision loss term is able to recover all of these inter-penetrations. Note that not even standard PBS is able to recover inter-penetrations between open meshes (cloth-to-cloth). Thus, again, PBNS appears to be more robust than PBS against collisions. Nonetheless, PBS is a more general solution. This property can be convenient for 3D artists, as it allows faster outfit design without hurting the final results. These results also show PBNS can handle cloth-to-cloth interactions accurately with minimal layer spacing. In the first sample, near the waist, we see up to three overlapping

FIGURE 5.7: PBNS results for outfits with overlapping layers of cloth. Each row shows a sample. From left to right: template outfit, posed sample without outermost layer, posed sample with whole outfit and cross section. The proposed collision loss can handle multiple layers of cloth, even when the template outfit presents inter-penetrations. As we can observe on the cross section, PBNS is able to generate consistent predictions with minimal layer spacing. Also note how it is possible to neurally simulate different fabrics –different wrinkle count and size– within the same outfit by assigning per-vertex weights to cloth related losses.

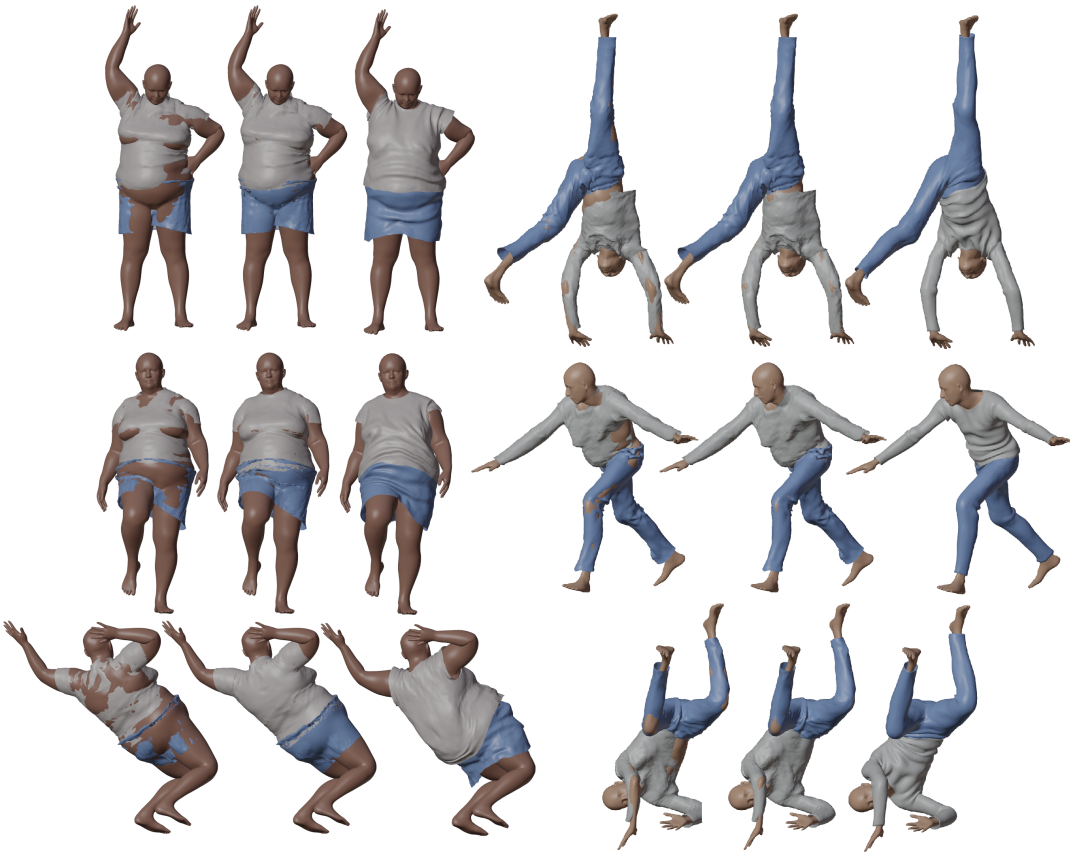


FIGURE 5.8: PBNS vs. TailorNet. We show two different bodies and outfits in three different poses each. For each pose, from left to right: TailorNet raw predictions, TailorNet after post-processing and PBNS. As observed, TailorNet heavily relies on post-processing, hurting performance and removing model differentiability. TailorNet predictions are noisy, show a bias towards body geometry and is unable to model cloth-to-cloth interactions. On the other hand, PBNS predictions show physically cloth-consistent predictions without body or cloth-to-cloth interpenetrations (raw predictions).

layers of cloth correctly sorted out –four layers if we consider the body. Additionally, different layers of cloth show different wrinkle count and size. This is due to the controllability of neural simulation parameters. Just like classic PBS, it is possible to simulate different fabrics by assigning per-vertex weights for $\mathcal{L}_{\text{edge}}$ and $\mathcal{L}_{\text{bend}}$. Higher weights will produce fewer, larger wrinkles (outer layers in the figure). Finally, the figure also shows the possibility to neurally simulate complements like gloves and boots. Rigid objects (boots) can be included into the neural simulation through parameter controllability (high weights). We are the first to propose a learning based methodology able to deal with cloth-to-cloth interactions, to allow result controllability by tuning parameters related to fabric physical properties –enhancing explainability– and defining a common framework for garments and complements.

5.5.5 Comparison

The current reference in the garment animation domain within the context of deep learning is TailorNet (Patel et al., 2020). We qualitatively compare results obtained with PBNS against TailorNet predictions. Note that TailorNet also addresses garment edition and resizing along animation. We choose two different body shapes

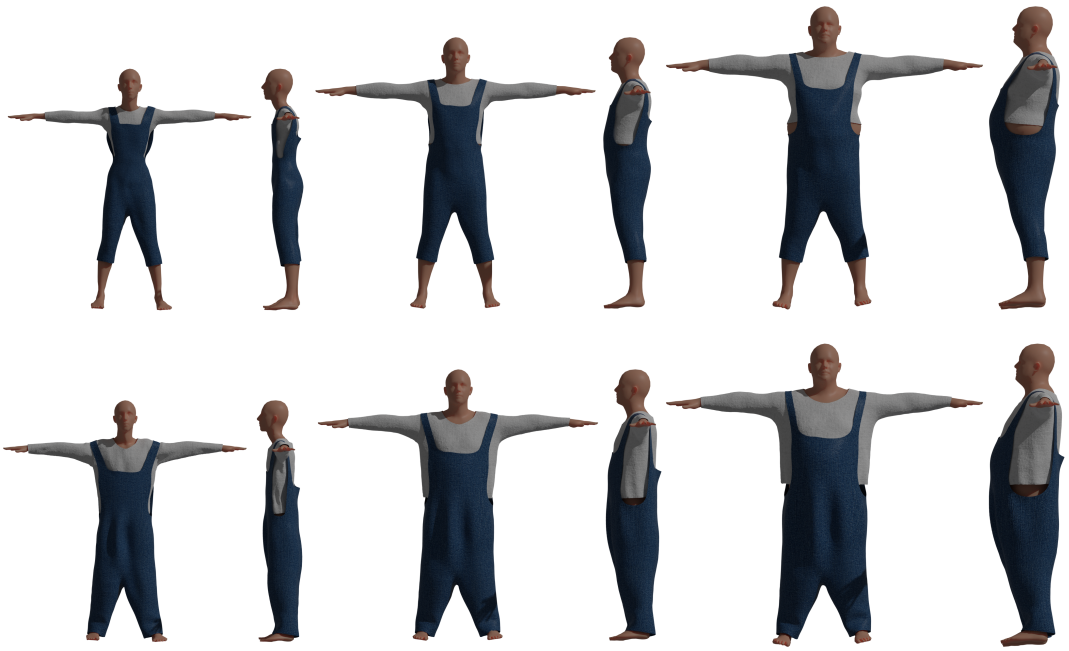


FIGURE 5.9: Results obtained using PBNS for outfit resizing. Body shape changes from left to right. Outfit tightness changes from upper to lower rows. All the previous properties discussed for PBNS are also present when it is used for outfit resizing.

and outfits and compare predictions obtained with both models. Authors of TailorNet post-process their predictions to solve collisions. For the sake of the comparison, since PBNS does not require post-processing, we show TailorNet with and without post-processing. Note that post-processing hugely increases computational time and removes model differentiability. Fig. 5.8 shows the results obtained. For each sample we show, from left to right: TailorNet raw predictions, TailorNet post-processed and PBNS. As can be seen, TailorNet predictions heavily rely on post-processing. TailorNet models individual garments independently, and it is thus unable to handle cloth-to-cloth interactions. On the other hand, PBNS can almost guarantee collision-free predictions even under extreme poses for cloth-to-body and cloth-to-cloth interactions. We can see that PBNS predictions are less noisy and better resembles cloth. Since TailorNet encodes garments as body offsets, body geometry is transferred to predictions for unseen body shapes (middle right and bottom right samples). We observe TailorNet quality diminishes as we move outside their training pose and shape distribution. Moreover, while both approaches are static (no dynamics), we observe TailorNet is unable to achieve temporal consistency, while PBNS does. In conclusion, while TailorNet compromises physical consistency and generalization to learn wrinkles, PBNS is able to generate pose-dependant wrinkles for completely unseen poses by imposing physical consistency. Additionally, PBNS complexity and computational time is several orders of magnitudes lower than TailorNet. TailorNet model size is in the order of gigabytes, while PBNS requires a few megabytes. Regarding computational time, for TailorNet we obtain around $3-5$FPS without post-processing and $0.6-0.9$FPS with post-processing, while PBNS can easily achieve hundreds or even thousands of executions per second (see Tab. 5.2).

FIGURE 5.10: PBNS is not limited to work with SMPL. Since it needs no PBS data, the engineering effort for the adaptation to custom avatars is minimum. Here we show different garments and outfits neurally simulated on top of custom avatars. For each, we show the avatar body in rest pose and two different viewpoints of a posed sample. PBNS can be used to enhance rigged 3D characters by animating their clothes.

5.5.6 Garment Resizing

In the current literature, we observe an interest on automatic garment resizing (Vidaurre et al., 2020; Tiwari et al., 2020; Patel et al., 2020). Current approaches rely as well on data, which requires gathering, labelling, formatting and other issues already discussed. We observe that, again, the problem can be solved unsupervisedly using a standard 3D animation format. We propose to use PBNS to automatically obtain 3D *animated* models –as blend shapes– that morph into different body shapes and sizes. Note that given a body shape, the resulting outfit model should be able to be retargeted to this body, but also change in size. We refer to this concept as *tightness*, while in previous works has been referred as *style* (Patel et al., 2020). To this end, we replace PBNS input θ with the body shape β concatenated with the garment tightness represented as $\gamma \in \mathbb{R}^2$. We also remove skinning. Finally, we compute an estimation for E_T in Eq. 5.4 transferring smoothed SMPL blendshapes by proximity. Fig. 5.9 contains the results obtained for a given outfit. We show three different body shapes (columns) and two different tightness (rows). As observed, PBNS allows resizing to the desired body shape and outfit tightness.

5.5.7 Custom Avatars

PBNS formulation is not dependant of SMPL, and thus, it can be applied to animate outfits on top of any rigged 3D animated model. We gather different avatars and poses from Mixamo[1], a free repository of 3D animated characters. We design or reuse outfits for the selected avatars. Then, we apply PBNS as described in this chapter. Since no PBS data is required, we obtain animated outfits for the avatars in a matter of minutes. Fig. 5.10 illustrates the results of this experiment. We show three different avatars. First, in rest pose, and then, two different viewpoints of the same pose. As can be seen, PBNS can be used to enhance rigged 3D characters by providing of realistic cloth behaviour to their outfits. This further proves the usefulness of the presented methodology for animated 3D character design.

5.6 Performance

We train our model on our subset of 2550 training poses with a batch size of 16 and Adam optimizer. We run our experiments on a GTX1080Ti. It takes around $1-2$ minutes per epoch, depending on the amount of collisions against the body. Using no GPU, it takes around $2-4$ minutes per epoch. Thus, training a model

[1] https://www.mixamo.com/

	Single	Batch
CPU	213 FPS	1235 FPS
GPU	455 FPS	14286 FPS

TABLE 5.2: PBNS performance. Tests done using an outfit with 12k vertices and 23.7k triangles, (the same as in Fig. 5.4). We run the trained model in CPU and GPU for single and batched samples. As observed, the chosen architecture is extremely efficient. No related work (deep-based or PBS) comes close to this level of performance.

using the methodology presented in this chapter does not require expensive hardware, enhancing accessibility for small film or videogame studios. Since there is no quantitative error, the stop criterion consists on qualitatively assessing validation predictions. It might take from 10 to 50 epochs to converge, depending on outfit complexity and body shape. During test, PBNS is extremely efficient. Tab. 5.2 shows the animation speed obtained with an outfit of 12k vertices and 23.7k triangles (same outfit as Fig. 5.4). We run the model in both, CPU and GPU, for single and batched samples. As can be seen, our methodology can generate over 14k samples per second. No previous work (deep-based or PBS) is near this level of performance. This is actually the expected behaviour, since PBNS yields skinned models with PSD. As aforementioned, these models are the standard for 3D animation and are designed to be extremely efficient. The only extra component is a small MLP that does not grow with vertex count. Since PBNS does not require post-processing, the reported numbers are the effective speed we would see in final applications. Additionally, due to the low size of the model, this solution is the only current methodology that can be applied in scenarios were computational resources must be available for other tasks, such as videogames and virtual reality, or in portable devices.

5.7 Conclusions, Limitations and Future work

We presented the first unsupervised deep learning based approach for outfit animation. More specifically, we described a methodology to neurally simulate outfits into blend shapes as Pose Space Deformations of Linear Blend Skinning models. Because of this, our solution is extremely efficient and can be easily integrated into any current 3D animation pipeline and run in almost any device (even low-computing or portable environments). We enabled unsupervised learning by formulating our problem as an implicit Physically Based Simulation. Furthermore, our proposed approach can be trained in a matter of minutes, even without a GPU. Therefore, the time from outfit design until model deployment is drastically reduced compared to previous approaches. PBNS can handle multiple layers of cloth, allowing neural simulation of complete outfits. Furthermore, we also show it can be easily adapted to any 3D avatar. This gives the methodology a broader applicability and higher scalability. CGI artists can design new animated draped characters more efficiently, and both, videogames and virtual try-ons, can easily introduce new 3D animated models for their outfit databases. Additionally, we presented the possibility of using this approach for automatic unsupervised outfit resizing. While the trained models do not present garment generalization as in Chapter 4, the many advantages outweigh this fact. That is, no need of costly data gathering, trained on a matter of minutes, more robust models thanks to unsupervised learning and a superior overall prediction quality. Additionally, while DeePSD can also handle complete outfits, PBNS explicitly learns cloth-to-cloth interactions.

In our approach, neither the input nor the potential energy formulation take into account the temporal dimension. This means that the learnt mapping from pose to mesh is unique. One can easily see how this pose-to-mesh uniqueness assumption does not hold by imagining simulating the same pose sequence at different speeds. A given pose $\boldsymbol{\theta}$ on the sequence shall produce different meshes $\mathbf{V}_{\boldsymbol{\theta}}$. This is specially important for neural simulation of very loose garments (dresses, long skirts, etc.). Such garments present a very dynamic behaviour that static approaches cannot reproduce. Thus, the presented methodology, as it is, is limited to quasi-static deformations and does not model cloth dynamics. Then, it works best for tight-fitting clothes. Chapter 6 explores including the temporal dimension in our neural simulator, while keeping its efficient formulation, to overcome the main limitations of PBNS.

Finally, also note that the idea of unsupervisedly learning to predict stable and physically consistent systems opens the possibility to generalize this methodology to handle other soft-tissue bodies. For example, hair, or human body self-collisions.

Chapter 6

Neural Cloth Simulation

FIGURE 6.1: We present a general solution for the garment animation problem through neural cloth simulation. More specifically, an unsupervised deep learning methodology inspired in physically based simulation. Ours is the first methodology able to learn cloth dynamics without any ground truth data.

6.1 Introduction

Cloth animation has been the focus of research during decades. This is mainly due to its numerous applications in the entertainment and fashion industry. First, computer graphics approaches that rely on physically based simulation to animate cloth (Baraff et al., 1998; Müller et al., 2007; Narain et al., 2012; Liu et al., 2013; Pfaff et al., 2014; Hahn et al., 2014; Macklin et al., 2016). While it is possible to obtain physically accurate results, these methodologies are computationally expensive and not suitable for scenarios where real-time is a requirement, such as video-games or VR/AR. The research community, inspired by the success of deep learning in other 3D tasks (Socher et al., 2012; Richardson et al., 2016; Qi et al., 2017; Han et al., 2017; Arsalan Soltani et al., 2017; Omran et al., 2018; Madadi et al., 2020) and its fast inference properties, has recently turned to neural networks as a suitable solution for a fast garment animation.

Initially, authors relied on supervision (Wang et al., 2019; Gundogdu et al., 2019b; Patel et al., 2020; Bertiche et al., 2020; Bertiche et al., 2021d). To simplify the problem, garments are usually skinned w.r.t. the underlying skeleton that drives the body motion. Then, the network task is to predict cloth deformations in rest pose.

These approaches present some drawbacks. Supervised learning requires huge volumes of computationally expensive data. Then, the process has to be repeated for each garment and body. Additionally, more often than not, data requires heavy pre-processing to be ready for training. Finally, predictions usually present body penetrations, which motivates the use of post-processing or strong regularization terms. Authors of (Bertiche et al., 2021c; Santesteban et al., 2022) identified these drawbacks and proposed unsupervised learning schemes. While these approaches addressed some drawbacks of supervised methodologies, they do not handle cloth dynamics. On one hand, PBNS (Bertiche et al., 2021c) uses a static formulation, and it is therefore unable to learn cloth dynamics. This methodology was previously presented in Chapter 5. On the other hand, SNUG (Santesteban et al., 2022) propose to use an inertia term from the computer graphics literature. Nonetheless, their adaptation of the inertia term to a deep learning framework only temporally smooths cloth particle velocities. We will see how this is not the same as learning cloth dynamics (Sec. 6.4.5).

We present the first methodology able to learn real cloth dynamics without the need of ground truth data. By doing so, we define the first general solution to neural garment simulation. The list of our contributions is as follows:

1. **Unsupervised Cloth Dynamics.** In the computer graphics literature we can find simulation based works that recast the equations of motion as an optimization problem. This means that a similar solution can be applied to deep learning. We adapt this solution to unsupervised training of neural networks. We will show how our methodology is the first to be able to learn cloth dynamics in an unsupervised fashion.

2. **Disentangled Cloth Subspace.** We analyze the nature of the garment animation problem to motivate a novel architecture that allows an automatic disentanglement of cloth static and dynamic deformations at a subspace level. We will see how this improves model performance as well as allowing controllability over cloth dynamics. Additionally, we leverage the disentanglement to propose a novel motion augmentation technique that further improves model generalization.

3. **In-depth Analysis on Neural Garment Simulation.** Unsupervised garment animation differs from other deep learning tasks, supervised or unsupervised. We provide of detailed analysis on the problem to understand its peculiarities and help to establish the bases of neural simulation for garments.

The rest of the chapter is as follows. In Sec. 6.2 we review the literature on cloth simulation and deep learning based methodologies on garments. Next, Sec. 6.3 describes the methodology we propose. Then, Sec. 6.4 contains an analysis on the different metrics, an ablation study and a comparison with the state-of-the-art. Finally, in Sec. 6.5 we discuss limitations and future research.

6.2 Related Work

Cloth simulation. Computer graphics has been tackling the cloth animation problem for decades now. The first advances in the field were done by (Weil, 1986), as static geometry based models. Later, researchers developed elastic continuum models for cloth (Feynman, 1986; Terzopoulos et al., 1987; Baraff et al., 1998; Carignan et

al., 1992) that permit dynamic simulations. On the other hand, other authors (Haumann, 1987; Breen et al., 1992; Provot et al., 1995) noted cloth is not a continuum, but a combination of mechanical interactions between cloth yarns. From here, alternative particle based formulations for cloth were developed, like the mass-spring model. Later, the work of (Baraff et al., 1998) presented triangle-based formulation for cloth that allowed fast simulation of complex garments. To this day, this work is still the foundation of many current methodologies for cloth simulation (Narain et al., 2012; Pfaff et al., 2014). Later, (Kaldor et al., 2008; Kaldor et al., 2010) proposed modelling cloth at yarn-level to achieve highly realistic behaviour. These methodologies –while accurate– are computationally expensive, therefore, not suitable for many applications that demand real-time performance. Simpler, more efficient formulations have been developed in favor of a faster simulation (Müller et al., 2007; Liu et al., 2013; Macklin et al., 2016) at the cost of accuracy or realism. Nonetheless, realistic simulation of fine cloth dynamics in real-time is still unfeasible with standard simulation. Specially as the number of cloth triangles increase and considering some of these real-time applications often require computational resources for other tasks. Subspace physics has proved a valid fast alternative for soft body simulation (Teng et al., 2014; Pan et al., 2015). This alternative has also been proposed for clothing (De Aguiar et al., 2010; Kim et al., 2013; Hahn et al., 2014), although in practice, collisions are not properly handled. Then, while computer graphics offers realistic and accurate solutions, efficient real-time cloth animation remains an open challenge.

Deep learning. During recent years, neural networks have proved their usefulness in many complex tasks. One of their main advantages is a fast inference time. Then, given their success in challenging 3D problems (Socher et al., 2012; Richardson et al., 2016; Qi et al., 2017; Han et al., 2017; Arsalan Soltani et al., 2017; Omran et al., 2018; Madadi et al., 2020), researchers have already turned to deep-based solutions for garment animation. Most of the current literature on the domain relies on supervised learning (Gundogdu et al., 2019b; Wang et al., 2019; Patel et al., 2020; Bertiche et al., 2020; Bertiche et al., 2021d; Santesteban et al., 2021). To this end, it is necessary to run hundreds or thousands of offline physics based simulations to gather the data required for training. Data gathering needs to be repeated for every garment, body and fabric parameters. This hurts the scalability of these solutions. Additionally, supervised learning is biased towards lower frequencies, yielding overly smooth garments. Moreover, supervision does not guarantee physical constraints are satisfied. Finally, simulators display a chaotic behaviour. Garment simulation on top of very similar body motions may result in considerably different outputs. This translates as noisy data, which hinders training. Authors of (Bertiche et al., 2021c) proposed PBNS, an alternative unsupervised solution that does not suffer from the drawbacks related to simulated data. To do so, they propose formulating physically based constraints as energy losses. This permits learning garment deformations without ground truth data. Nonetheless, their formulation is purely static, meaning they do not handle cloth dynamics. Similarly, SNUG (Santesteban et al., 2022) uses the same unsupervised scheme as PBNS to learn garment deformations. With the only addition of an inertia loss term from the physics based simulation literature (Liu et al., 2013; Martin et al., 2011; Gast et al., 2015) to model dynamic deformations. However, in this work we will see how their adaptation of the inertia term is not modelling true cloth dynamics. Following the trend of unsupervised learning for garment animation, we present the first work able to learn cloth dynamics without *ground truth* data. Thus, defining the first general solution for neural cloth simulation. Moreover, we propose a novel model architecture that automatically

disentangles static and dynamic cloth deformations. This, in turn, shows improved performance and interesting novel properties.

6.3 Methodology

The goal of this work is to define a deep-learning based methodology for the garment animation problem. This problem corresponds to the animation of cloth draped around skinned 3D body models. Following the current trend (Bertiche et al., 2021c; Santesteban et al., 2022), we propose an unsupervised training inspired on physically based simulation. We additionally achieve a disentanglement of static and dynamic cloth deformations by considering the nature of both cases in our model design.

6.3.1 Neural Cloth Subspace Solver

Defining the problem. Our methodology for learning cloth dynamics unsupervisedly is inspired in classical computer graphics physical simulation (Baraff et al., 1998; Müller et al., 2007; Liu et al., 2013). To this end, cloth is presented as a particle system $\mathbf{x} \in \mathbb{R}^{N\times 3}$. Cloth solvers compute the cloth configuration at instant t from the previous cloth state, which is defined by the particle locations and velocities at $t-1$. Additionally, the solver must also consider external forces –colliders, wind, etc– that act on the cloth. Within the scope of this work –garment animation– the external forces correspond to collisions with the underlying skinned 3D model draped with the clothes and gravity, which is constant. Then, given a skinned 3D model parametrized by its pose and location $\boldsymbol{\theta}$, the solver can be written as:

$$\mathbf{x}_t = f(\boldsymbol{\theta}_t, \mathbf{x}_{t-1}, \mathbf{v}_{t-1}), \tag{6.1}$$

where $\mathbf{v}$ is the particle velocities. Velocities will depend on the current and previous particle locations, therefore, we can rewrite the expression as $\mathbf{x}_t = f(\boldsymbol{\theta}_t, \mathbf{x}_{t-1}, \mathbf{x}_{t-2})$. Likewise, we have that $\mathbf{x}_{t-1} = f(\boldsymbol{\theta}_{t-1}, \mathbf{x}_{t-2}, \mathbf{x}_{t-3})$. The same could be done for $\mathbf{x}_{t-2}$, $\mathbf{x}_{t-3}$ and so on and so forth. This yields the following:

$$\mathbf{x}_t = \hat{f}(\boldsymbol{\theta}_t, \boldsymbol{\theta}_{t-1}, \boldsymbol{\theta}_{t-2}, ..., \boldsymbol{\theta}_0, \mathbf{x}_0). \tag{6.2}$$

Thus, assuming the 3D body model parametrized by $\boldsymbol{\theta}$ and the gravity are the only external forces, the cloth can be fully parameterized by the body pose history –or motion– $\boldsymbol{\Theta}_t = \{\boldsymbol{\theta}_t, \boldsymbol{\theta}_{t-1}, \boldsymbol{\theta}_{t-2}, ..., \boldsymbol{\theta}_0\}$. We can intuitively assume that early poses will have less impact on the current cloth state. This allows to safely discard poses outside a given temporal window. To ensure the problem remains well-posed, the temporal window size should be sufficiently large. Garments that may show more complex, longer dynamics –like dresses or skirts– will require a larger temporal window size than garments that will not show complex dynamics –like tight pants. We also identify separately the static case, a special case of garment animation in which there is no body or cloth motion. This corresponds to the result of draping a body staying still in a given pose during an *infinite* –long enough– amount of time. For such case, we have $\mathbf{x}_t = \mathbf{x}_{t-1} = ... = \mathbf{x}_0$ and $\boldsymbol{\theta}_t = \boldsymbol{\theta}_{t-1} = \boldsymbol{\theta}_0$, hence, the cloth can –and must– be fully parametrized using only $\boldsymbol{\theta}_t$.

Cloth subspace. Once we have been able to establish a relationship between the body motion space and cloth space, we are implicitly declaring that a clothing subspace $\mathcal{Z} \subset \mathbb{R}^d$ exists (with $d \ll N \times 3$). That is, $\boldsymbol{\Theta}_t \rightarrow \mathbf{z}_t \rightarrow \mathbf{x}_t$, with $\mathbf{z}_t \in \mathcal{Z}$.

Then, inspired on subspace physics (De Aguiar et al., 2010; Kim et al., 2013; Hahn et al., 2014), our proposed model must be able to solve the next cloth configuration from the current subspace encoding:

$$\mathbf{z}_{t+1} = g(\boldsymbol{\theta}_{t+1}, \mathbf{z}_t), \tag{6.3}$$

where $g(\cdot)$ corresponds to a neural cloth subspace solver. Additionally, it must be designed in a way that ensures that given a static sample –no motion– the subspace encoding should not change, $\mathbf{z}_t = \mathbf{z}_{t-1} = ... = \mathbf{z}_0$. The optimal solution must naturally fall back to a static formulation when there is no input body motion. We know the static solution to be a special case of the general problem, likewise, the subspace of all static cloth states $\mathbf{z}^S \in \mathcal{Z}^S$ is a subspace of the subspace of all possible cloth states, $\mathcal{Z}^S \subset \mathcal{Z}$, in the same way that the body pose space is a subspace of the body motion space. That is:

$$\boldsymbol{\theta}_t \rightarrow \mathbf{z}_t^s, \quad \boldsymbol{\Theta}_t \rightarrow \mathbf{z}_t. \tag{6.4}$$

We know the pose space to be a single continuous manifold, henceforth, due to eq. 6.4, the static cloth subspace must also be a single continuous manifold. Similarly, we can extend this reasoning to the motion space and the full cloth subspace. We can then conclude that subspace $\mathcal{Z}$ is a higher-dimensional manifold around the static subspace $\mathcal{Z}^S$.

Neural network. Motivated by all of the exposed, we propose an encoder-decoder recurrent neural network as a suitable architecture for this problem. The model takes body motion $\boldsymbol{\Theta}_t$ as input, encodes it as $\mathbf{z}_t$ and finally decodes it into the predicted cloth $\mathbf{x}_t$. Also, we present a novel disentangled encoder that will enforce –by design– the expected static and dynamic behaviour. We will show how this results in an improved performance and explainability. Moreover, by disentangling static and dynamic deformations, we allow controllability on the level of motion in our predictions. This property will help artists to achieve the desired looks for a given application. We also define a novel motion augmentation technique that greatly increases model robustness. As it is usual in the literature (Gundogdu et al., 2019b; Bertiche et al., 2020; Bertiche et al., 2021d; Patel et al., 2020; Bertiche et al., 2021c; Santesteban et al., 2022), the network predicts cloth deformations in rest pose. Then, the garment is skinned w.r.t. the underlying body skeleton and it is posed along the body mesh.

6.3.2 Body Motion Descriptors

As explained, to fully parametrize the garment state $\mathbf{x}_t$ we need the body pose history, which to we refer as body motion $\boldsymbol{\Theta}_t$. To keep the problem tractable, we safely truncate the pose history using a reasonable temporal window size. Window size will be directly related to the maximum cloth motion length that our network will be able to learn. Looser garments –like skirts– require longer temporal windows. For the rest of the chapter, we will refer to the truncated pose history as $\boldsymbol{\Theta}_t = \{\boldsymbol{\theta}_t, \boldsymbol{\theta}_{t-1}, \boldsymbol{\theta}_{t-2}, ..., \boldsymbol{\theta}_{t-n}\}$. Then, during training, we predict each sample $\mathbf{x}_t$ using $\boldsymbol{\Theta}_t$ as network input. For inference, the model can be fed with indefinitely long sequences, as new poses $\boldsymbol{\theta}$ are used to update the hidden recurrent state.

The naive baseline solution would be to feed body pose sequences –as joint orientations– along the global velocity. While this would suffice to avoid an ill-posed problem, this representation is sub-optimal and dynamics are entangled with static information. The model would need to learn by itself to extract body motion information from the input poses. This increases the required training data and time, as

well as model capacity, while hurting generalization. We propose a set of disentangled descriptors more suitable for this problem.

Static descriptors. To describe the body pose it is common to use joint relative orientations (relative to the parent joint). The usual axis angle or quaternion representations suffer from a many-to-one problem and discontinuities in the rotation space. Thus, we opt for the 6D descriptors proposed in (Zhou et al., 2019). Then, we observe relative orientations are local descriptors. Garment deformations depend on the global body configuration. Small changes in the first joints of the kinematic tree can lead to significantly different garment states. Therefore, samples close to each other in the input space would need to be mapped to points very far from each other in the output space, which makes training and generalization more challenging. On the other hand, using global joint orientations would make the input space extremely large and noisy. Rotations around the gravity axis would create completely different inputs, while the output should remain the same. We propose using for each joint –besides the local 6D descriptor– a unit vector with the *unposed* direction of the gravity. That is:

$$\hat{\mathbf{g}}_j = \mathbf{R}_j^{-1}\mathbf{g}/g, \tag{6.5}$$

where j is the joint index, $\mathbf{R}$ is the rotation matrix corresponding to the global joint orientation, $\mathbf{g}$ is the gravity vector and $g = 9.81\text{m/s}^2$. This descriptor contains information about the global orientation of each joint and it is invariant to rotations around the gravity axis. Additionally, it will be correlated with the direction of local cloth deformations –in rest pose– due to gravity. We concatenate our local and global descriptors, yielding a 9-dimensional feature array per joint, that is, $\boldsymbol{\theta}^S \in \mathbb{R}^{K\times 9}$ where K is the number of joints.

Dynamic descriptors. To describe body motion we take the derivatives in time of the joint orientations and locations. Orientation derivatives are computed from the static descriptors explained in the previous paragraph. Then, location derivatives are computed from the joint locations in space. Note that these derivatives would suffer from the same issues as the global joint orientations, i.e. a large and noisy input space. We address the issue by *unposing* derivatives as in eq. 6.5 without normalizing them. This greatly reduces the input space as well as defining a descriptor that it is more strongly correlated to the local cloth dynamic deformations due to motion, both in magnitude and direction. We assume no air resistance, therefore, dynamic cloth deformations will appear only when the body is under acceleration. Hence, we use as motion descriptors the first derivative of joint orientations –any rotation implies an acceleration– and the second derivative of the joint locations. Both descriptors are concatenated into a 12-dimensional descriptor per joint, which gives us $\boldsymbol{\theta}^D \in \mathbb{R}^{K\times 12}$.

Skinned 3D models have often many joints in hands, feet and face which are unlikely to be relevant for garment dynamics. We remove these joints from the input.

6.3.3 Model

In this section we present our model architecture. As explained, we propose a recurrent encoder-decoder network architecture. Our encoder is composed of two different modules, a static and a dynamic encoder, each fed with the corresponding descriptors. Both encodings are combined by addition to be later decoded into local cloth deformations. Finally, the garment is posed along the body. Fig. 6.2 depicts our model.

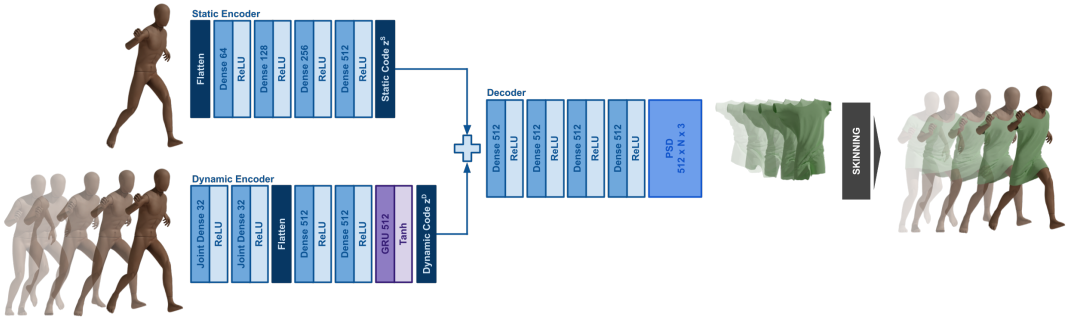


FIGURE 6.2: Model architecture. We design our model as a recurrent encoder-decoder. The input of the model is the body motion as described in Sec. 6.3.2. The encoder is disentangled into a static a dynamic encoder, each fed with the corresponding descriptors. Dynamic encoder layers have no bias, ensuring a direct correlation between input motion and dynamic activations. Encodings are combined by addition and decoded into local cloth deformations. Finally, the garment is skinned along the body.

Static encoder. We implement this encoder as a set of 4 fully connected layers. The encoder is fed only with the current pose $\boldsymbol{\theta}_t^S$, which is flattened first into a $9K$-dimensional array. The output of the encoder is a static latent code $\mathbf{z}_t^S$.

Dynamic encoder. This module is implemented in two blocks. A set of fully connected layers and a Gated Recurrent Unit (GRU). First, 2 fully connected layers are applied to per-joint descriptors individually –as if joints were samples– to obtain a high-level feature array per joint. We empirically observed this to be beneficial since dynamic descriptors are a concatenation of different modalities. Later, the array is flattened and fed to an additional 2 fully connected layers. Finally, the output is passed through the GRU, which combines it with its hidden state –that encodes the history of dynamics– to obtain the dynamic latent code $\mathbf{z}_t^D$. Note that $\mathbf{z}_t^D$ is computed with the whole motion $\boldsymbol{\Theta}_t$. We observe adding multiple GRUs makes training unstable and hurts model inference speed. All layers of the dynamic encoder have no bias. Without bias, zero input translates to zero output (this is not necessarily true the other way around). This ensures that a static sample will have a null $\mathbf{z}_t^D$, since time derivatives –dynamic descriptors– will be zero. Thus, addition with $\mathbf{z}_t^S$ will have no impact. Furthermore, samples with high body motion will generally produce high values for $\mathbf{z}_t^D$, creating high dynamic deformations due to a high perturbation of $\mathbf{z}_t^S$. Additionally, the hidden state of the GRU will fade away to zero as long as new poses with no motion are being fed to the model. This means that the latent code $\mathbf{z}$ will naturally fall back to $\mathbf{z}^S$ when motion stops. This also guarantees that, if no motion is present, neither $\mathbf{z}$ nor the output will change, as it will depend only on $\boldsymbol{\theta}_t$. It would not be possible to ensure this with an entangled encoder or biases in the dynamic branch. Finally, this design allows padding sequences with still frames without altering the value of $\mathbf{z}_t^D$. This property is convenient during training.

Separate encoders have an additional advantage. Their respective input spaces have less variability. Because of this, their latent codes will be more meaningful and they will generalize better. An entangled encoder would need to learn an input space with combinations of static and dynamic descriptors. This would make the task more challenging.

6.3.4 Training

We train our model unsupervisedly by applying losses inspired in physically based simulation, following the trend of (Bertiche et al., 2021c; Santesteban et al., 2022). Losses are implemented as energy functions of the physical system composed by cloth and body. As training progresses, the network learns to predict garment states that satisfy the energy constraints.

Cloth Model. In the computer graphics literature we can find multiple ways of defining cloth models. From a simple mass-spring model to more sophisticated continuous approaches that compute triangle deformation energies (Baraff et al., 1998; Liu et al., 2013; Narain et al., 2012). To be able to use a cloth formulation within a deep learning framework, the only requirement is to be differentiable. This makes our approach compatible with most cloth models, allowing them to be freely interchanged. For this work, we implemented mass-spring model as in (Bertiche et al., 2021c), a squared version of the continuum formulation of (Baraff et al., 1998) and Saint Venant Kirchhoff elastic material model as in (Santesteban et al., 2022). As a loss $\mathcal{L}_{\text{cloth}}$, this term will penalize in-plane deformations.

Bending Loss. We implement our bending term for out-of-plane deformations as the squared difference of the angle between adjacent faces w.r.t. the angle in the rest garment (Pfaff et al., 2014). Then, for each pair of adjacent faces:

$$\mathcal{L}_{\text{bending}} = k_b \frac{l^2}{8a}(\phi_t - \phi^R)^2, \tag{6.6}$$

where k_b is the bending stiffness, l is the length of the common edge, a is the summation of the area of both faces, ϕ_t is the dihedral angle and ϕ^R is the dihedral angle in the rest garment. With this formulation, the garment will try to retain its original shape. Scaling the loss as a function of the edge length and triangles area makes it agnostic to mesh resolution and connectivity.

Collisions. In computer graphics simulations, cloth interaction with external objects is obtained by detection and solving of collisions. Similarly, we implement a loss term that penalizes collisions and creates repelling gradients, pushing cloth vertices outside the body (Bertiche et al., 2021d; Bertiche et al., 2021c):

$$\mathcal{L}_{\text{collision}} = k_c \min(\mathrm{d}(\mathbf{x}_t; \boldsymbol{\theta}_t) - \epsilon, 0)^2, \tag{6.7}$$

where k_c is a balancing factor, $\mathrm{d}(\cdot; \boldsymbol{\theta})$ is the signed distance to a body mesh parametrized by $\boldsymbol{\theta}$, with negative values inside, and ϵ is a small threshold to ensure robustness.

Inertia Loss. Following the laws of motion, a moving object will retain its velocity unless forces act on it. Thus, differences in the location of the cloth particles $\mathbf{x}_t$ and the projected location obtained with the previous velocity $\mathbf{x}_t^{\text{proj}} = \mathbf{x}_{t-1} + \mathbf{v}_{t-1}\Delta t = 2\mathbf{x}_{t-1} - \mathbf{x}_{t-2}$ are due to other acting forces. A similar observation has already been made in the context of simulation (Liu et al., 2013; Martin et al., 2011; Gast et al., 2015). This led to the possibility of obtaining the next garment state by finding the critical points of the following expression:

$$h(\mathbf{x}_t) = \frac{1}{2}(\mathbf{x}_t - \mathbf{x}_t^{\text{proj}})^T \mathbf{M}(\mathbf{x}_t - \mathbf{x}_t^{\text{proj}}) + \Delta t^2 E(\mathbf{x}_t), \tag{6.8}$$

where $\mathbf{M}$ is the mass matrix of the particle system, Δt is the time step of the simulation and $E(\cdot)$ is the potential representing internal and external forces. This leads to

the following loss term for each particle:

$$\mathcal{L}_{\text{inertia}} = \frac{1}{2\Delta t^2} m(x_t - x_t^{\text{proj}})^2, \tag{6.9}$$

where m is the particle mass. Since $\mathbf{x}_t^{\text{proj}}$ depends on $\mathbf{x}_{t-1}$ and $\mathbf{x}_{t-2}$, we need to run the model for $\mathbf{\Theta}_{t-1}$ and $\mathbf{\Theta}_{t-2}$ as well. It is crucial not to back-propagate gradients through $\mathbf{x}_{t-1}$ and $\mathbf{x}_{t-2}$. Otherwise, while the loss value gets lower, the model will not show true cloth dynamics. The reason for this is that $\mathbf{x}_t$ would have influence in the location of $\mathbf{x}_{t-1}$ and $\mathbf{x}_{t-2}$ by generating pulling or pushing gradients. Thus, information would be travelling back in time. Note how simulation based related works from which we extract this loss term do not optimize $\mathbf{x}_{t-1}$ or $\mathbf{x}_{t-2}$ to satisfy eq. 6.8. One important observation is that this loss will penalize differences in velocities. Thus, whenever the underlying body skeleton joints present no rotations or accelerations, there will not be accelerations in the cloth –since it is attached by blend weights to the skeleton– and gradients from this term will be zero. Then, no dynamic deformations would be necessary to satisfy the loss. This is consistent with the explanation in Sec. 6.3.2 regarding the choice of dynamic descriptors.

Gravity. As previous unsupervised approaches, we include the effects of gravity by implementing its potential energy as a loss:

$$\mathcal{L}_{\text{gravity}} = -\mathbf{M}\mathbf{x}_t\mathbf{g}. \tag{6.10}$$

This term will push vertices in the direction of the gravity, weighted by particles mass and gravity.

6.4 Results

In this section we explain the results obtained with the proposed methodology. First, we describe the data that we use, as well as the experimental setup. Next, we define the different metrics that we use to evaluate our methodology along with a discussion on how to interpret them. Then, we explore the impact of different cloth material models, followed by an ablation study where we analyze the value of each contribution. Later, we compare our methodology with the current state-of-the-art. Finally we show a novel motion control property that arises thanks to our proposed disentangled architecture.

6.4.1 Experimental setup

To run our methodology for a given skinned 3D body model we need a dataset of pose sequences. To do so, we gather a few 3D avatars and pose sequences from Mixamo[1]. The motions include a few tens of variations for different kind of actions: walking, running, jumping, turning and spinning. Note that some motions contain combinations of these actions. This totals around 450 motion sequences, containing around 45000 poses. For each action, we use 5% for validation and 10% for test. Additionally, in order to compare with state-of-the-art methodologies –PBNS (Bertiche et al., 2021c) and SNUG (Santesteban et al., 2022)– we use AMASS dataset (Mahmood et al., 2019) for SMPL model (Loper et al., 2015a). For fairness, to allow comparison against SNUG public checkpoint, we train PBNS public code and our methodology on the same data. While poses are discrete in time, we implement

[1]https://www.mixamo.com/

a continuous sampling by using Slerp (Shoemake, 1985). This allows training our methodology with arbitrary time steps Δt. During training, samples $\mathbf{\Theta}$ are batched. For each sequence $\mathbf{\Theta}_t$, we predict the garment for the last 3 time instants to compute $\mathcal{L}_{\text{inertia}}$. As explained, it is crucial to back-propagate the inertia loss only through $\mathbf{x}_t$. While the static loss terms can be *safely* applied to all 3 predictions, we observe this creates a sampling bias that hurts performance.

Balancing terms of each loss are related to desired fabric properties. For the cloth term, the values will depend on the chosen cloth material model. Usually within the range $[5, 15]$ for structural stiffness and $[0, 1]$ for shearing. For bending, we use values in the range $[1\text{e}{-5}, 1\text{e}{-4}]$. For collisions, we set k_c to a value similar to the chosen structural stiffness and a threshold $\epsilon = 4mm$. Higher values for k_c will compromise other metrics without improving generalization. Collision-free predictions will depend mostly on training data distribution. Particle mass m –or mass matrix $\mathbf{M}$– is computed from vertex area –from the garment mesh representation– and the chosen fabric surface density. Then, the temporal window will depend on the looseness of the garment. We go from 0.5 seconds for tighter garments up to 2 seconds for looser garments. Training times until convergence will differ greatly depending on garment, body and motions. From 1 hour for simpler garments up to a day for garments that show complex and rich dynamics. Inference is extremely fast, as we show in Tab. 6.4. We refer to the supplementary code for additional details. The code will be publicly released.

6.4.2 Metrics

Traditionally, specially for supervised approaches, lower values for error metrics indicate better predictions. This is not the case for this specific unsupervised problem. Furthermore, the metrics will behave differently for the static case. For that reason, it must be considered and evaluated separately.

Cloth Model. This metric must be related to the cloth material model used during training. For mass-spring, edge elongation is the most suitable. For continuous formulation, the strain or triangle deformation is a better choice. This metric will measure in-plane cloth deformations. Cloth is resistant to stretching forces, thus, lower values are desired in this case. Some formulations allow modelling shearing forces separately. Cloth is not resistant to shearing, and thus, lower values do not necessarily imply better predictions.

Bending. Measured as the average error on dihedral angles w.r.t. rest angles for each pair of adjacent faces. Cloth is not resistant to bending. Lower is not necessarily better. A higher bending stiffness will reduce this error, but generate different wrinkles. Thus, a lower error does not imply better predictions, but a stiffer fabric. Note that a null bending error would only be achieved by the template garment in rest pose. Nonetheless, extremely high values for this metric might suggest a failed simulation. Ultimately, this property must be assessed qualitatively.

Collisions. Expressed as the percentage of vertices placed within the body. Collisions are to be avoided. For this metric, lower values are desired. For the dynamic case this metric shows an interesting behaviour in the validation data. First, it rapidly decreases until a minimum. Afterwards, as the network learns to predict cloth dynamics, the metric slightly increases until it plateaus. This behaviour does not appear using a static formulation.

Gravity. Computed as the potential energy of the predicted garments. This energy is defined relative to an arbitrary 0. Then, it is not possible to define a *goal* for this metric. Its value will also depend on the garment, fabric density, 3D body

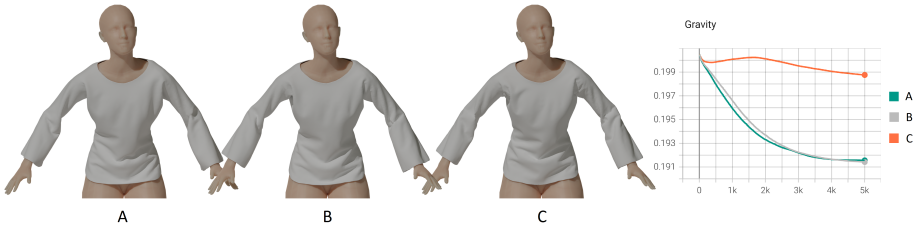


FIGURE 6.3: Comparison of cloth material models. Our methodology is compatible with any differentiable formulation for cloth. We test and compare three different ones as a static optimization problem: A) mass-spring model as in (Bertiche et al., 2021c), B) the continuous formulation proposed by (Baraff et al., 1998) and C) the Saint Venant Kirchhoff material used in (Santesteban et al., 2022). For static optimization, we can use gravity as a measure of convergence.

and pose data. **Static case:** the optimal solution is achieved when the garment has reached an equilibrium state. For this case, given the same aforementioned experiment conditions, lower is better. Other metrics must be considered as well. A lower cloth stiffness would allow further stretching in the direction of gravity. In such case, we could not conclude the approach converges to a more optimal solution. On the other hand, we can state that training converged when this metric plateaus. **Dynamic case:** adding motion to the problem changes the behaviour of this metric. For example, a spinning skirt will raise against gravity when dynamics begin to appear, increasing the value of this metric. Likewise, jumping sequences will make the garment *float* when the falling motion begins. This means the garment is not always at its lowest position. For this reason, it is not possible to conclude that lower values are better. Usually, the static formulation gives lower values for gravity. Therefore, a dynamic model with lower gravity metric might be due to a lack of cloth dynamics.

Inertia Loss. Measured as the error between $\mathbf{x}_t$ and $\mathbf{x}_t^{\text{proj}}$ weighted by the particle mass. As explained in Sec. 6.3.4, lower values do not translate into true cloth dynamics. We empirically observe it behaves the other way around. As training advances and cloth dynamics are being learnt, the value of this metric increases. On the contrary, under the static formulation, this metric will usually decrease. Similar to how gravity metric indicates model convergence for the static case, this metric does the same for the dynamic case. Convergence in training but divergence in validation shows overfitting. This metric also gives an intuition of the level of motion in the predictions. The value of this metric and its evolution during training will greatly depend on garment, fabric, body and motion data.

6.4.3 Cloth Model

We analyze the effect of different cloth material models. To this end, we simulate in a static fashion the same garment, body and poses with different cloth models. We test a mass-spring formulation used by authors of (Bertiche et al., 2021c), the continuous formulation of (Baraff et al., 1998) and the Saint Venant Kirchhoff (StVK) elastic material model as in (Santesteban et al., 2022). We adapt the material model of (Baraff et al., 1998) by squaring their constraints for stretching and shearing. We depict the obtained results in Fig. 6.3. As shown, the chosen cloth formulation has almost no impact on the qualitative result. Nonetheless, we observe some differences worth mentioning. As opposed to continuous formulations, mass-spring fabric parameters depend on mesh resolution and connectivity. On the other hand, we find that StVK formulation has much more difficulties achieving convergence. We can

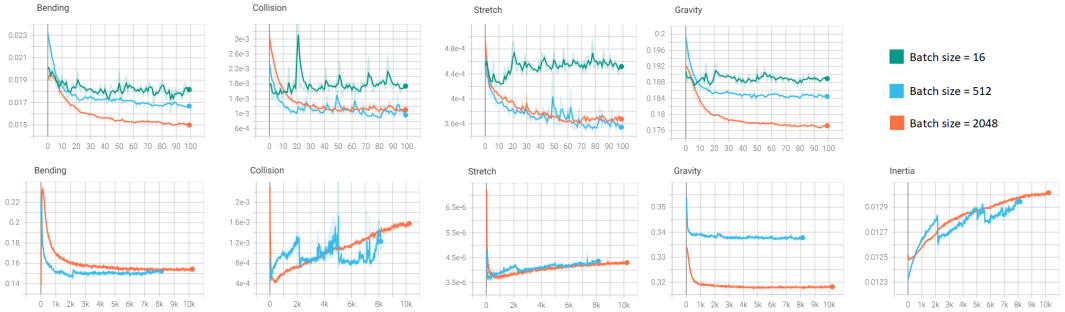


FIGURE 6.4: Training progress for different batch sizes. The upper row corresponds to static experiments. The lower row corresponds to dynamic experiments. Using bigger batch sizes significantly increases convergence of the predictions. On the dynamic problem, bigger batches result in a more stable training.

see this in the gravity plot adjoined to Fig. 6.3. This formulation may be sub-optimal within an optimization framework. Finally, the formulation of (Baraff et al., 1998) allows explicit control of shearing stiffness. We observed each cloth model scores lower in their respective strain –as expected– and thus, it is not useful to compare strains quantitatively.

6.4.4 Ablation

Batch size. Unsupervised garment animation is quite sensitive to batch size. The reason for this is as follows. Model evolution during training is similar to physical simulation. In the static case, each input sample $\boldsymbol{\theta}_i$ has a theoretical optimal output $\mathbf{x}_i$. Under supervised training, gradients in the output will point directly towards $\mathbf{x}_i$, with stronger magnitudes for more erroneous predictions. This is not the case for unsupervised garment animation. Gradients will try to greedily update the output cloth by pointing to an intermediate state $\hat{\mathbf{x}}_i$, similar to how a simulation would compute intermediate states until achieving the fully converged solution $\mathbf{x}_i$. Moreover, there is no guarantee that $\hat{\mathbf{x}}_i$ will be closer in space to $\mathbf{x}_i$ than previous predictions. The *path* the model has to follow to convergence is not straight-forward. Furthermore, gradient magnitudes cannot be used as a measure of convergence, except in the extremely unlikely case that their value is 0 for all the samples in our dataset. On top of that, we have to consider special constraints found in deep learning. Network updates, specially for small batches, can *undo* the work of previous iterations. Then, the network gets stuck in sub-optimal local minima. This produces stiff garments with wrinkling patterns that repeat across different poses. With dynamics, the problem becomes more complex, since the gradients for $\mathbf{x}_t$ depend on $\mathbf{x}_{t-1}$ and $\mathbf{x}_{t-2}$, which are also predictions. Fig. 6.4 shows training metrics for static and dynamic problems with different batch sizes. For the static case, we see that increasing batch size greatly improves model convergence. For the dynamic case, we see that using a small batch size makes the training noisy. There is no plot for batch size 16 for dynamics since small batches make training unstable and usually fails.

Static descriptors. We study the impact of the proposed static descriptors under a static formulation. This means training without $\mathcal{L}_{\text{inertia}}$. We test three different static descriptors. Tab. 6.1 contains the results obtained. The first row corresponds to raw pose data as quaternions or axis angle representations. Next, 6D descriptors as in (Zhou et al., 2019). Finally, the static descriptors proposed in Sec. 6.3.2. The proposed descriptors present fewer collisions, showing better generalization. They

	Strain (mm)	Bending (rads)	Collision (%)	Gravity
Raw	0.35463	0.0168	0.087418	0.1845
6D	0.37374	**0.01372**	0.10778	0.1811
6D+G	**0.35145**	0.01475	**0.073881**	**0.1765**

TABLE 6.1: Static descriptors ablation. We run experiments for each static descriptor under a static formulation. First, raw joint orientation data as quaternions or axis angles. Second, 6D descriptors as in (Zhou et al., 2019). Finally, our proposed static descriptors. Our approach shows better generalization and convergence without compromising cloth integrity.

also achieve higher convergence, since gravity is lower. Finally, we see also a lower strain value, which means it did not compromise cloth integrity to minimize the other metrics.

Dynamic descriptors. We test three different alternatives as motion descriptors. First, as baseline, we use body pose and root joint velocity. This descriptor is the minimum requirement to avoid an ill-posed problem. Second, we gather joint rotation speed and accelerations without *unposing* (full-space). Finally, our proposed descriptors, after *unposing* joint accelerations (local-space). Fig. 6.5 depicts the training plots for collisions and inertia. We omit other metrics since their values barely differ. At left, we have collision metric. At right, inertia metric. The first row corresponds to training set plots, and the lower row to the validation set. It is interesting to notice that training plots (top row) are almost the same. For validation (bottom row), we observe full-space descriptors show more collisions. These descriptors have a higher variability. Thus, it is a much more challenging task for the network to learn the whole input space. This results in worse generalization. The other two descriptors are local, then, validation set distribution is more likely to fall in the same distribution learnt by the network. For the inertia metric, training and validation plots –although differ in absolute values– show similar curves. We see the baseline features diverge from the other curves. This indicates a lower generalization. The model has difficulties in extracting meaningful motion information from the baseline representation. Providing of explicit joint rotation speeds and accelerations helps the network to better learn the dependency between body motion and dynamic cloth deformations.

Architecture. We perform an analysis on model architecture by comparing our disentangled approach against a single encoder. For this experiment, the *entangled* encoder is fed with static and dynamic descriptors. Next, the GRU receives the output encoding. Finally, the decoder predicts garment deformations. The result of this comparison is shown in the first two rows of Tab. 6.2. First row corresponds to a single *entangled* encoder. The second row corresponds to the proposed disentangled architecture. We see that separate encoders have better generalization –lower *strain* and *collisions*– than a single encoder. Additionally, disentangled encoders make training much more stable. Training with an entangled encoder usually fails. This justifies our motivations in the network design.

Temporal window. As presented in eq. 6.2, predictions need of the pose history. We analyze the effect of different temporal window sizes. For this experiment, we neurally simulate a t-shirt with a window size of 0.1 seconds, 0.5 seconds and 1 second. We present the metrics of this experiment in Fig. 6.6. As we can see from the *inertia* metric, a smaller window gives a lower level of dynamics. During training, a reduced input data has a direct impact on the discriminative power of the network. The model cannot properly differentiate motions that are similar within the

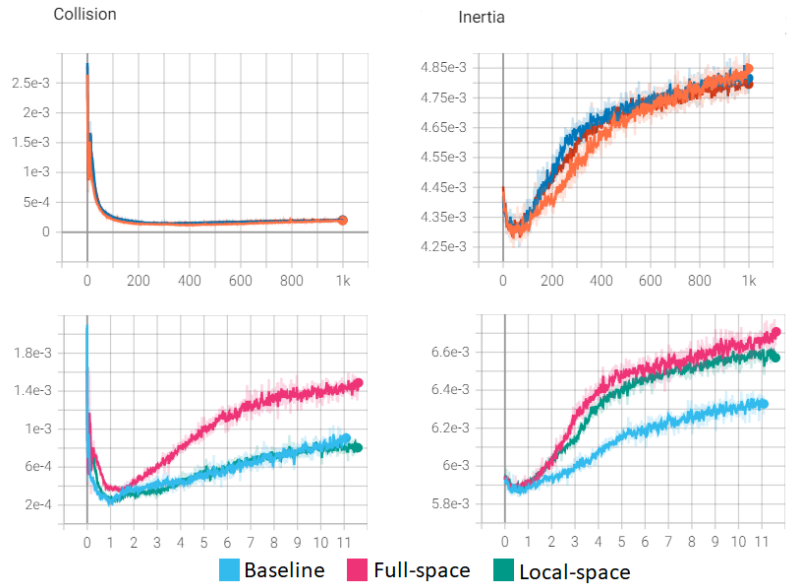


FIGURE 6.5: Ablation on dynamic input features. Baseline features contain body pose and root joint velocity. The other experiments contain body pose and joint rotation speeds and accelerations. For the local-space features, we *unpose* accelerations. We evaluate collision (left) and inertia (right) metrics. Upper row corresponds to training data. Lower row to validation data. We see local-space features generalize better –fewer collisions– while showing a similar level of dynamics as the training data –same inertia curves. We omit color labels for training plots since they are almost the same.

	Strain	Collisions (%)	Gravity	Inertia
Entangled	9.7840	1.101	1.068	1.889
Disentangled	8.2391	0.694	1.066	1.891
+Mirror	8.2146	0.505	1.067	1.864
+Aug. motions	**7.6241**	**0.323**	1.068	1.905

TABLE 6.2: Ablation study on network architecture and data augmentation. Experiments with + contain the *improvements* from the previous rows. First row: single encoder. Second row: disentangled encoder. Third row: pose mirroring. Last row: motion augmentation. First and second experiment have no data augmentation. Second, third and fourth experiment have the same architecture. We see how a disentangled encoder and the proposed data augmentations have a beneficial impact. Additionally, we see *gravity* and *inertia* show no significant difference. Therefore, these improvements do not compromise other cloth properties.

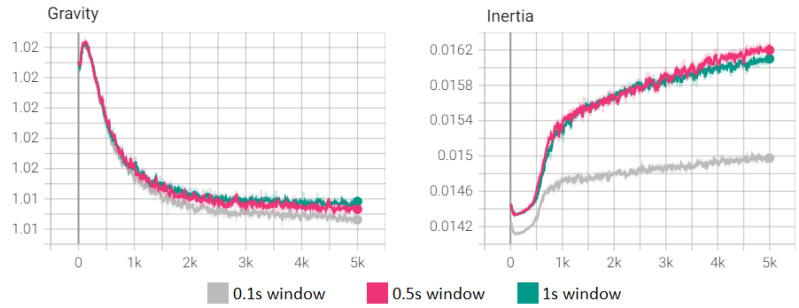


FIGURE 6.6: Ablation on temporal window size. We compare neural simulation metrics for a t-shirt with a temporal window size of 0.1 seconds, 0.5 seconds and 1 second. We can see the level of dynamics –*inertia* metric– is much lower with a small temporal window. We can also observe a slight decrease in *gravity* metric in this case. This is an example of the effect of dynamics on gravity explained in Sec. 6.4.2. On the other hand, we see that further increasing temporal window size has almost no impact in the results. We omit other metrics since their values are almost equal.

0.1 second window, but have a different past. Because of this, predictions converge to the average of the motions that are close to each other in this reduced input space, which lowers the level of dynamics. Additionally, we observe a lower value for the *gravity* metric for the smaller window size. This is related to the discussion on the effect of dynamics on gravity in Sec. 6.4.2. On the contrary, we observe an even larger temporal window size has no significant impact in cloth dynamics –for this specific case– but increases the required VRAM and time for training. Therefore, for each specific garment, body and motions, it is important to find a window size that achieves a compromise between dynamics and efficient training. Note that during inference, sequence length can be arbitrarily large. We additionally test the effect of Δt by training at different frame rates, 15 and 60 (for all other experiments the frame rate is 30). We notice the model is able to learn realistic cloth dynamics even at lower frame rates. Furthermore, at 15 fps training is significantly faster. On one hand, the amount of samples is halved. On the other hand, for the same temporal window, the length of the pose sequences is smaller, which means less operations. Finally, the gradients generated by $\mathcal{L}_{\text{inertia}}$ are larger, and dynamics take less time to appear. On the contrary, increasing the frame rate makes the task much more challenging for the same reasons (but opposed). Nonetheless, training at higher frame rate allows learning finer cloth dynamics.

Augmentation. As the model learns cloth dynamics, collisions in the validation set increase. We study the possibility of mitigating this effect with data augmentation. On one hand, we use standard pose mirroring with probability of 50%. On the other hand, leveraging our disentangled approach, we devise a novel motion augmentation technique. To do so, during training, we shuffle the dynamic latent code $\mathbf{z}^D$ for a portion of the samples from each batch (20% in this experiment). We can apply only the static loss terms to the *augmented* samples. We do not back-propagate gradients to the encoders for these samples. This will give the decoder more data points from $\mathcal{Z}$ and around $\mathcal{Z}^S$, increasing generalization. Tab. 6.2 shows the effect of the different augmentations. Second row, no augmentation. Third row, pose mirroring. Last row, motion augmentation. We see that for both augmentations, *strain* and *collision* metrics are noticeably reduced, specially for motion augmentation. Moreover, we see it almost completely mitigates the increase of collisions as cloth dynamics are being learnt. This is shown in Fig. 6.7, left plot (*collisions)*. It is also important to notice that *gravity* and *inertia* metrics show very little difference. This means the

FIGURE 6.7: Data augmentation ablation. *Collision* (left) and *inertia* (right) metrics evolution for validation set during training for different augmentation techniques. First: no augmentation. Next: pose mirroring. Finally, a novel motion augmentation technique. The latter is only possible thanks to disentangled encoders. As cloth dynamics are being learnt by the network, collisions slightly increase. Pose mirroring reduces this effect, while motion augmentation mitigates it almost completely. We also see the evolution of cloth dynamics –*inertia* metric– is not compromised by these changes.

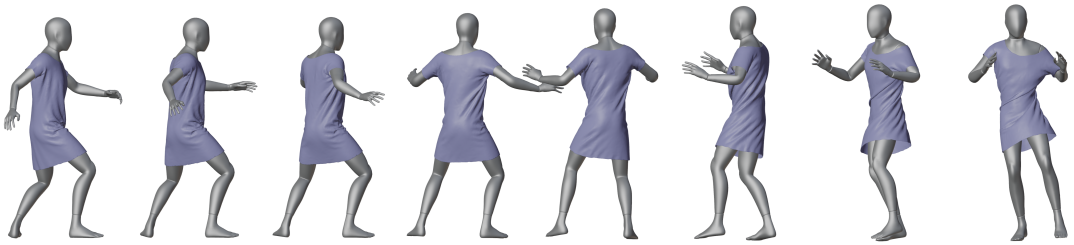


FIGURE 6.8: Sample sequence. We illustrate predictions for a dress during a spinning motion. It can be seen how the dress coils up the mannequin body as it spins.

generalization improvement does not compromise other properties.

We show qualitative results of our methodology for different garments, bodies and motions in Fig. 6.1, 6.8 and 6.9. Fig. 6.1 shows results for SMPL model with different body shapes and the *mannequin* model from Mixamo. The samples show rich and meaningful cloth dynamics. Fig. 6.8 depicts predictions for a dress during a spinning sequence. As observed, as the motion begins, the dress coils up the mannequin body as we would expect to happen. Finally, in Fig. 6.9 we can see how our methodology can generalize to different articulated 3D bodies and garments.

6.4.5 State-of-the-art Comparison

We evaluate our methodology against recent unsupervised approaches for garment animation. On one hand, a quasi-static solution, PBNS (Bertiche et al., 2021c). On the other hand, a methodology that claims to model dynamic cloth deformations, SNUG (Santesteban et al., 2022). SNUG authors provide of a checkpoint we use for comparison. Nonetheless, their methodology is adapted to work with the body shape variability of SMPL. For fairness, we also implement and train their methodology –based on their public code and paper– using constant body shape. Tab. 6.3 shows quantitative results for each model. Qualitative evaluation is included in Fig. 6.10. We show samples for different body motions: a) jumping, b) leap forward,

FIGURE 6.9: Qualitative samples. Our methodology is compatible with any articulated 3D body and garment. Here we show predictions obtained after neural simulation of different garments draped on different bodies.

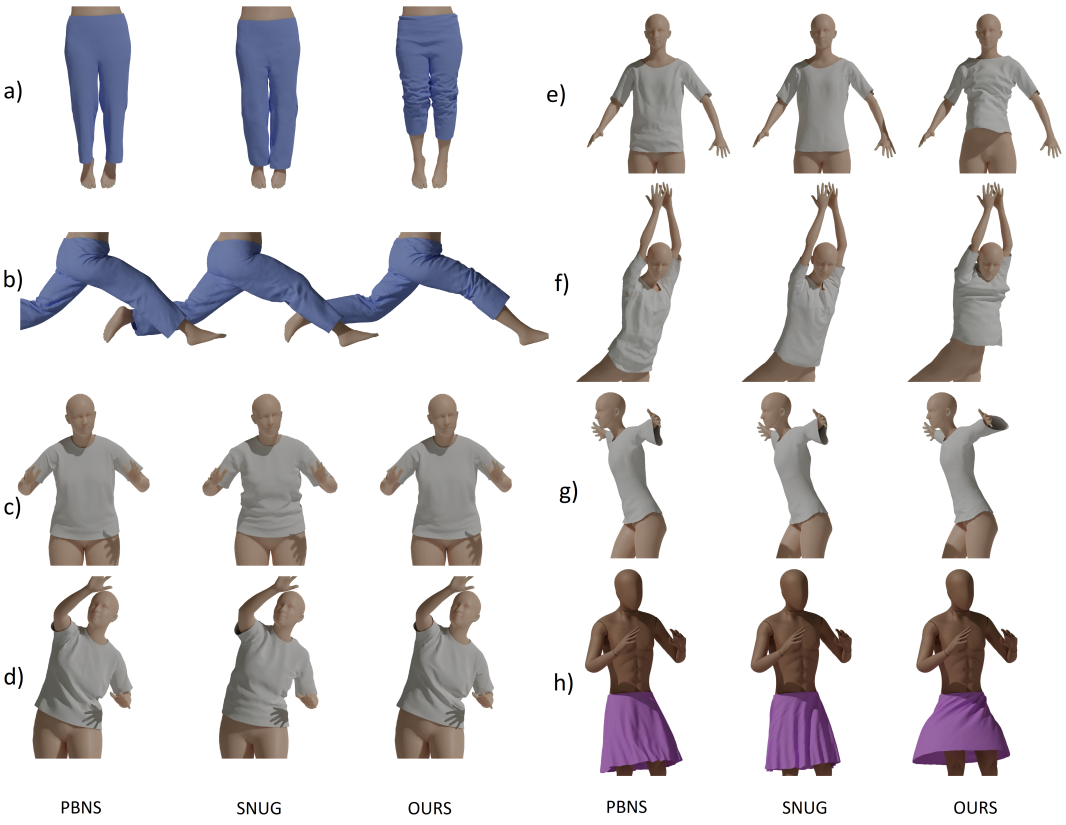


FIGURE 6.10: Qualitative comparison for different kind of motions. The motions are: a) jumping, b) leap forward, c) quasi-static pose, d) quasi-static pose, e) jumping, f) dancing jump, g) flapping arm motion and h) fast spin. We use PBNS public code. For samples *a*, *b*, *c* and *d*, we use SNUG public checkpoint. For samples *e*, *f*, *g* and *h*, we implement and train SNUG based on authors public code and paper. Since PBNS uses a static formulation, it shows no cloth dynamics. Then, we see SNUG appears to be unable to show any meaningful dynamics. This is mainly due to an incorrect implementation of the inertia loss. Additionally, looking at the quasi-static samples (*c* and *d*), we notice wrinkling patterns that repeat for most poses. This gives a stiff and less realistic look. Finally, our approach shows dynamic cloth deformations consistent with body motion.

	Strain	Bending	Collision (%)	Gravity	Inertia
PBNS	3.18	0.20	0.274	0.6274	0.937
SNUG	3.89	0.20	0.283	0.6299	0.553
Ours	4.2	0.15	0.276	0.6403	1.219

TABLE 6.3: Quantitative comparison with state-of-the-art: PBNS (Bertiche et al., 2021c) and SNUG (Santesteban et al., 2022). SNUG achieves a much lower value for the inertia term. Nonetheless, see in Fig. 6.10 that their approach shows no cloth dynamics. As explained in Sec. 6.4.2, lower values do not translate to cloth dynamics.

c) quasi-static pose, d) quasi-static pose, e) jumping, f) dancing jump, g) flapping arm motion and h) fast spin. We use PBNS public code. We know PBNS to be a static solution and therefore, as expected, it does not show dynamic cloth deformations. It is interesting to notice that for quasi-static samples, our solution and PBNS converge to a similar garment. This is the expected behaviour, since our network design ensures the model naturally falls back to a static formulation when there is no input motion. For SNUG, samples *a*, *b*, *c* and *d* are obtained with their checkpoint. Samples *e*, *f*, *g* and *h* are obtained with our implementation of SNUG. We observe SNUG does not show any meaningful cloth dynamics. After analyzing SNUG, we notice their approach is unable to learn dynamics by design. First, authors observe that $\mathcal{L}_{\text{inertia}}$ depends on $\mathbf{x}_t$, $\mathbf{x}_{t-1}$ and $\mathbf{x}_{t-2}$ and assume that it is possible to train with sub-sequences of only 3 body poses. From eq. 6.2 we know the whole body motion is needed. During SNUG training, $\mathbf{x}_{t-1}$ and $\mathbf{x}_{t-2}$ are computed from $\{\boldsymbol{\theta}_{t-1}, \boldsymbol{\theta}_{t-2}\}$ and $\{\boldsymbol{\theta}_{t-2}\}$ respectively. This is severely ill-posed, thus, their estimation of $\mathbf{x}_t^{\text{proj}}$ will be poor. In practice, this means their model will be unable to learn any motion longer than ~ 0.0666 seconds (3-frame time span for 30 FPS sequences). On the other hand, based on their public code and results, we observe they incorrectly back-propagate $\mathcal{L}_{\text{inertia}}$ through $\mathbf{x}_{t-1}$ and $\mathbf{x}_{t-2}$. This will give lower values for the loss but not true cloth dynamics. See in Tab. 6.3 how their inertia metric is the lowest by far. This is even more noticeable in looser garments. See the skirt sample *h* under a fast spinning motion in Fig. 6.10. For circular motions, we have an acceleration $a = \omega^2 r$. Back-propagating $\mathcal{L}_{\text{inertia}}$ through previous frames –as SNUG– will modify particle locations on all frames to minimize accelerations. Then, their implementation generates gradients pointing inwards that *close* the skirt to minimize a by minimizing r. On the other hand, our implementation minimizes the distance between $\mathbf{x}_t$ and $\mathbf{x}_t^{\text{proj}}$, without modifying the latter. For circular motions, particle velocities are tangential to the circumference. This means that $\mathbf{x}_t^{\text{proj}}$ will always be outside this circumference. Our approach generates gradients pointing outwards that *open* the skirt to minimize the distance to $\mathbf{x}_t^{\text{proj}}$. In their paper, SNUG authors mention that training on longer sequences resulted in lower inertia values but not true dynamics. Their approach is not learning cloth dynamics, it is smoothing cloth particle velocities. This may give the illusion of a slight cloth dynamic deformation for very specific motions and garments –in some jumping motions– but it will fail for the general case, as we show in all other motions. See supplementary video for additional comparison. Because of these reasons, we cannot consider that SNUG is able to learn cloth dynamics. Additionally, in samples *c* and *d*, we see wrinkling patterns repeating. This happens across most poses. This gives a stiff and unrealistic look to the cloth. Because SNUG is trained with a small batch size, it suffers from the issue explained in Sec. 6.4.4.

We additionally compare running times of the different methodologies. In the

	PBNS	SNUG	Ours
FPS	7.5K	15	853.9

TABLE 6.4: Comparison of running times against state-of-the-art: PBNS (Bertiche et al., 2021c) and SNUG (Santesteban et al., 2022). Due to its simple formulation and no temporal dimension, PBNS is the fastest approach by far. Then, due to the use of pre-computed data from the body, multiple GRUs and the need of post-processing, we see that SNUG struggles to achieve a reasonable performance. Our approach, while it does not achieve the efficiency of PBNS, can easily run faster than real-time.

FIGURE 6.11: Motion control. In this image we show a still frame of a spinning motion. Thanks to the disentanglement of static and dynamic cloth deformations achieved by the network design, it is possible to control the level of motion in the cloth. To do so, we scale the latent dynamic code to linearly interpolate –and extrapolate– from the static subspace to the full subspace. Leftmost to middle samples: linear interpolation from static latent code $\mathbf{z}^S$ to latent code $\mathbf{z}$. Middle to rightmost samples: linear extrapolation.

literature we find authors that consider only the forward pass of the network as running time. This is misleading. We measure the time it takes to compute the final prediction along the underlying body from the raw pose data. This gives a much more accurate idea of the running times to be expected on final applications. We report the results in Tab. 6.4. PBNS achieves the fastest performance by far due to their simple formulation and no need to model temporal dimension. For SNUG, we use the checkpoint and code provided by its authors. Their model struggles to keep 15 frames per second for garments with only 4K vertices. We see that SNUG heavily depends on post-processing to solve collisions. To this end, it needs expensive precomputed data for each sample that makes it much slower. Finally, while our approach does not achieve the performance offered by PBNS, it runs significantly faster than real-time. We run all performance tests in a machine with AMD Ryzen 7 5800H and a RTX3060.

6.4.6 Cloth Subspace Disentanglement and Motion Control

The methodology presented in this work is designed to automatically learnt disentangled subspaces for static and dynamic cloth deformations. To prove our model is effectively doing so, we linearly interpolate and extrapolate between the static cloth subspace $\mathcal{Z}^S$ and the full cloth subspace $\mathcal{Z}$. To this end, we scale the dynamic latent code given by the dynamic encoder $\mathbf{z}^D$ with $w \in [0, 2]$. That is, $\mathbf{z} = \mathbf{z}^S + w\mathbf{z}^D$. This can be seen in Fig. 6.11. From left to right, the values for w go from 0 to 2 with steps of 0.2. The sample in the middle is the standard network output. This is learnt automatically by the network without any sample or latent code manipulation.

6.5 Conclusions and Limitations

We presented a general methodology for unsupervised garment animation. Contrary to previous related works, our work is the first methodology able to learn cloth dynamics without ground truth data. Additionally, we devise a novel disentangled architecture that improves generalization, opens the possibility of a new motion augmentation technique that greatly increases robustness and allows for motion controllability, which is a useful never seen before property for artists. Also, we provide of detailed analysis and insights on neural cloth simulation that will help future research on the domain. We proved the effectiveness of our methodology with different 3D avatars and garments. In Chapter 5 I introduced PBNS, a novel methodology for unsupervised garment animation. Due to its simplicity, efficiency, robustness and qualitative results, it is the first viable option for garment animation in real-life applications. Nonetheless, it is limited to quasi-static deformations. In this chapter a overcome this limitation and present a complete solution for the unsupervised neural garment animation problem for the first time. While this compromised the performance of the model (compared to PBNS), it is still largely faster than real time.

Removing the need of gathering ground truth data is a huge advantage in the domain, already noted by previous works (Bertiche et al., 2021c; Santesteban et al., 2022). Nonetheless, simulation cannot be completely skipped. Instead of being an offline preprocessing, simulation and training are now the same. The first time the network sees a given training sample, its corresponding $\mathbf{x}^{\text{proj}}$ will be an estimation implicitly computed from body motion (transferred to the cloth through garment blend weights). The next epoch, this estimation will be more accurate. So on and so forth. This means that fine cloth dynamics take time to appear, since they have to be indeed simulated within the network. It would not be possible for the network to learn these dynamics without going through these intermediate states. This effect is even greater for looser garments. This means that supervised training will always be much faster, even without considering that unsupervised losses are more computationally expensive. Nonetheless, the methodology still has a significant advantage over supervised approaches, since similar sample motions will contribute to each others neural simulations. On the contrary, even for very similar motions, data gathering through simulation has to be done from scratch for every sequence. Another advantage of unsupervised training is that the network will converge to the simplest solution. That is, similar motions will show similar deformations. On the other hand, offline simulations may show a huge variability for similar motions. Because of this, supervised training needs to deal with noisy data that makes the task much more challenging. Finally, supervised training shows a bias towards lower frequencies and does not satisfy physical constraints without explicit regularization. Then, overall, unsupervised training is still much less time-consuming than data gathering through simulations and will generally converge to simpler, more robust models. We believe an interesting line of research for the future is to study the possibility to kickstart neural simulation with sparse simulated data. Afterwards, finetuning by neural simulation will permit training on an arbitrary number of motions with the desired fabric parameters.

Chapter 7

Conclusions

Within this thesis I portrayed my personal adventure through the wilderness of the research world and into the deep learning domain in pursuit of an efficient and realistic 3D garment animation. I look back and I can contentedly state I laid an additional brick on the stairs leading to the desired goal, on top of the ones previously placed by innumerable researchers, and below the countless ones that are yet to come. Hopefully, not long from now, I will be able to behold the fruit of my labour implemented as part of some popular film or videogame. On the meantime, I will review the results obtained through this thesis. First, I will analyze the outcome of each of the chapters, this will give an idea of how much has been achieved. Next, I will highlight what is missing, that is, how far the research community still is from the final goal. Finally, I will give my intuitions regarding promising research lines that may help achieve what is missing.

In the first chapter I described a methodology for human pose and shape recovery from still RGB frames. This method relies on an intermediate 3D representation that consists not only on body joints, but also in landmarks placed across the body surface. As opposed to other works, this allows unambiguous prediction of body shape and the orientation of some specific joints. The results show an improvement with respect to related works. I noticed 3D pose and shape estimation quickly gained popularity and the domain became saturated, with new methods showing marginal improvements. At this point, I put my focus on a less explored topic within human-centric scenarios, 3D clothing. At first, with the goal of complementing 3D human regression from images with garments. As a next step, in the following chapter, I introduced CLOTH3D dataset. I created CLOTH3D motivated by the scarcity of publicly available data. At the time it was clear how 3D garment by itself is already a challenging task. Then, I shifted my aim from computer vision towards 3D vision. In this direction, the dataset is accompanied of a baseline generative model for 3D garments. This baseline, trained on CLOTH3D, allows generation of a great variety of garment types conditioned on human pose and shape. On one hand, I note cloth simulation on arbitrary human shapes and poses is very challenging. There are many factors that may lead to corrupted results, invalidating the data. Additionally, cloth quality is not fully faithful to its real-life counterpart, specially for complex cloth-to-body interactions. Then, despite I was able to publish a large dataset with thousands of valid sequences thanks to manual supervision, unfortunately, data generation for garments is not a straightforward process that can be easily automated pressing a button. On the other hand, from the generative model, I observe that the common registration process of garments against the body is prone to lose geometric details and diminishes the overall quality of the garments. Also, trying to design a single model able to generate such a large number of garment types for any shape and pose is too ambitious. Nonetheless, it is worth noting that it was possible

to learn a meaningful shared latent space for all the garments. The latter observations on cloth modelling with deep learning led to the next chapter, DeePSD. For this method, I focus on garment animation only instead of jointly with generation. In this new formulation of the problem, I assume I have the *template* 3D garment in rest pose as input. This approach yields higher quality results. Additionally, it allows enforcing priors on the cloth surface based on the template garment. Moreover, it is the first solution able to handle multiple garments, overlapping pieces of cloth and generalize to completely unseen outfit meshes. Furthermore, with efficiency in mind, I formulated the network such that it does not predict garments directly, but a set of blend weights and blend shapes. This is the standard format for 3D animated models. Thus, during test, this methodology achieves real-time trivially (hundreds or thousands of executions per second). In spite of this, I observed how different garments, bodies, materials and other factors led to significantly different wrinkles and cloth deformations. It is extremely hard to learn such fine details for thousands of different garments using a single model. Then, DeePSD tends to show coarser wrinkles that are common for most garments. Cloth behaviour is not satisfying yet. I also observed this in related works, where the quality of the garments also has room for improvement. Noticing this, I reduce even further the focus of my research. No garment generation, and now, no garment variability anymore. The goal is to obtain efficient and realistic garment animation for a single outfit per network. To this end, it would be necessary to collect exhaustively samples for the chosen outfit in many different poses, which I know it is specially challenging. Inspired by the previous priors I successfully applied in DeePSD training, I design a fully unsupervised loss function. I explore this solution in the PBNS chapter. I combine this loss with a simple Multi-Layer Perceptron. Surprisingly, the results greatly outperform qualitatively any other previous work. The outfits present realistic cloth behaviour, without over stretching or compression, fine coherent wrinkles and barely no collisions, all of this even under extreme unseen poses. I extended the collision term of the loss to explicitly learn, for the first time, cloth-to-cloth interactions. Robust animation of full outfits with multiple layers of cloth is now a reality. On top of that, training takes only a few tens of minutes, almost completely excusing the outfit-specific nature of the solution. Finally, thanks to the simple architecture, the methodology achieves an entirely unprecedented level of performance. All of the characteristics that PBNS show make it the first viable deep learning based solution for real-life applications. PBNS implies a leap forward in the quality of garment animation in real-time applications. The simplicity of its formulation and how easily it can be trained places the solution at the reach of even small videogame or film studios. PBNS is a strong step towards the goal of this thesis. There is a drawback. PBNS is limited to quasi-static deformations. Looser garments that show rich dynamics cannot be properly modelled with this solution. They show a stiff and unrealistic behaviour. In the final contribution, Neural Cloth Simulation, I explicitly address this issue. I propose another unsupervised formulation that is additionally able to learn realistic dynamics. The obtained results show an unparalleled garment quality and dynamics compared to other deep learning solutions. I also introduce a novel architecture design and input descriptors that permit automatic disentanglement of static and dynamic garment subspaces. Noticing the peculiarities of the domain, I provide of detailed analysis of it to guide future research. I show how subspace disentanglement has interesting applications, such as a novel motion augmentation technique for training and a never-seen-before motion control property, which might prove useful for 3D artists. Despise the increased complexity of the problem, the proposed solution is still able to achieve real-time performance easily with a small memory footprint and without

expensive hardware requirements. Neural Cloth Simulation is another significant step towards the goal of this thesis.

At first, I tried *solving* the 3D garment problem as a whole, generation and animation of arbitrary garments, body shapes and poses. While potentially possible, it was too ambitious at the time. Then, first I removed the generation part. Next, garment and body shape generalization. I focused on designing methodologies for efficient and realistic animation of single outfits. By the end of this thesis, I obtained results comparable with Physically Based Simulation. As next steps, a first relevant challenge for neural garment animation is self-collision. Commonly, computer graphics approaches use the garment *history* to avoid self-collisions. An initial state with no self-collisions and careful simulation allows solving them before they appear. As of now, it is not possible to guarantee valid initial states in deep learning. Suitable history-free algorithms for self-collision solving are necessary. Once this is a reality, it would be time to go back, step by step, to the initial problem, without compromising quality nor efficiency. For PBNS I successfully explored body shape generalization. Intuitively, Neural Cloth Simulation can use as input any body parameterization (such as body shape and/or pose). Body shape generalization would mean more training time with a bigger model. Unhandled, this may make Neural Cloth Simulation a feasible solution for large 3D studios only. Actual progress is achieved only when everyone can benefit from it. In this regard, the methodology would greatly improve by kickstarting dynamics, perhaps with sparse supervision or a better formulation. The next interesting step is garment generalization, while it is also the most complex. The reason for this is that garments show strong global correlations. Pick up a piece of cloth by two opposite corners and let it hang a bit. Now, pull one of its corners. This will instantly change the whole geometry of the cloth. In practice, this means different garments may show completely different deformations under the same poses, even if the only difference is the location of the seams. Fortunately, in this domain, accuracy is not the priority, but looks. There is no need for a neural network to show the exact same wrinkles a given garment would show in real life, only plausible ones. In this sense, a possible shortcut could be to learn the building blocks of the garments, like a set of *eigenwrinkles*, and learn how to place them to satisfy the physical constraints imposed by the underlying body and cloth material. The final step I envision would be a complete generalization to any piece of cloth and collider (body or other objects). This would be a neural simulator (not to confuse with neural simulation). Whether this is possible or not is not the challenge. The challenge is to beat modern simulation algorithms, which get faster every day. There are three possible way to do so: less operations per vertex, more efficient collision handling and larger time steps. Can a neural network compute a *simulation* step with less vertex operations than modern simulators? Many simulators present a bottleneck at the collision solving step, would it be possible for neural networks to simplify this task while generalizing to any collider? Finally, simulators often require hundreds of steps per second, could a neural network simulate large time steps ($\Delta t = 1/30$s) in a stable, realistic and efficient way? It is unlikely for neural networks to reduce the number of operations per vertex and step. On the other hand, standard collision solving is a complex issue because it is hard to parallelize (to leverage GPU computational power). A neural based solution would not face this issue. Finally, even if a neural network requires more operations per step, this would be compensated by using large time steps. It is not unrealistic to believe neural networks could do so with higher stability than standard simulation. Overall, neural cloth animation is still in its infancy and only time will tell how tall it will grow, and as a player or a fan, I hope to see it myself.

Bibliography

Alldieck, Thiemo et al. (2018a). "Detailed human avatars from monocular video". In: *2018 International Conference on 3D Vision (3DV)*. IEEE, pp. 98–109.

— (2018b). "Video based reconstruction of 3d people models". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 8387–8397.

Alldieck, Thiemo et al. (2019). "Tex2shape: Detailed full human body geometry from a single image". In: *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2293–2303.

Allen, Brett, Brian Curless, and Zoran Popović (2002). "Articulated body deformation from range scan data". In: *ACM Transactions on Graphics (TOG)* 21.3, pp. 612–619.

Amberg, Brian, Sami Romdhani, and Thomas Vetter (2007). "Optimal step nonrigid ICP algorithms for surface registration". In: *2007 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, pp. 1–8.

Anguelov, Dragomir et al. (2005). "SCAPE: shape completion and animation of people". In: *ACM TOG*. Vol. 24. 3, pp. 408–416.

Arsalan Soltani, Amir et al. (2017). "Synthesizing 3d shapes via modeling multi-view depth maps and silhouettes with deep generative networks". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1511–1519.

Atrevi, Dieudonné Fabrice et al. (2017). "A very simple framework for 3D human poses estimation using a single 2D image: Comparison of geometric moments descriptors". In: *Pattern Recognition* 71, pp. 389–401.

Bailey, Stephen W et al. (2018). "Fast and deep deformation approximations". In: *ACM Transactions on Graphics (TOG)* 37.4, pp. 1–12.

Bailey, Stephen W et al. (2020). "Fast and deep facial deformations". In: *ACM Transactions on Graphics (TOG)* 39.4, pp. 94–1.

Baraff, David and Andrew Witkin (1998). "Large steps in cloth simulation". In: *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pp. 43–54.

Barber, Elizabeth Jane Wayland (1991). *Prehistoric textiles: the development of cloth in the Neolithic and Bronze Ages with special reference to the Aegean*. Princeton University Press.

— (1995). *Women's work: the first 20,000 years: women, cloth, and society in early times*. WW Norton & Company.

Bertiche, Hugo, Meysam Madadi, and Sergio Escalera (2020). "CLOTH3D: Clothed 3D Humans". In: *European Conference on Computer Vision*. Springer, pp. 344–359.

— (2021a). "Deep Parametric Surfaces for 3D Outfit Reconstruction from Single View Image". In: *2021 16th IEEE International Conference on Automatic Face and Gesture Recognition (FG 2021)*, pp. 1–8. DOI: 10.1109/FG52635.2021.9667017.

— (2021b). "Neural Implicit Surfaces for Efficient and Accurate Collisions in Physically Based Simulations". In: *CoRR* abs/2110.01614. arXiv: 2110.01614. URL: https://arxiv.org/abs/2110.01614.

Bertiche, Hugo, Meysam Madadi, and Sergio Escalera (2021c). "PBNS: Physically Based Neural Simulation for Unsupervised Garment Pose Space Deformation".

In: *ACM Trans. Graph.* 40.6. ISSN: 0730-0301. DOI: 10.1145/3478513.3480479. URL: https://doi.org/10.1145/3478513.3480479.

Bertiche, Hugo et al. (2021d). "DeePSD: Automatic deep skinning and pose space deformation for 3D garment animation". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 5471–5480.

Bhatnagar, Bharat Lal et al. (2019). "Multi-Garment Net: Learning to Dress 3D People from Images". In: *Proceedings of the IEEE International Conference on Computer Vision*, pp. 5420–5430.

Bogo, Federica et al. (2016). "Keep it SMPL: Automatic estimation of 3D human pose and shape from a single image". In: *ECCV*, pp. 561–578.

Brau, Ernesto and Hao Jiang (2016). "3d human pose estimation via deep learning from 2d annotations". In: *3D Vision (3DV)*. IEEE, pp. 582–591.

Breen, David E, Donald H House, and Phillip H Getto (1992). "A physically-based particle model of woven cloth". In: *The Visual Computer* 8.5, pp. 264–277.

Bronstein, Michael M et al. (2017). "Geometric deep learning: going beyond euclidean data". In: *IEEE Signal Processing Magazine* 34.4, pp. 18–42.

Carignan, Michel et al. (1992). "Dressing animated synthetic actors with complex deformable clothes". In: *ACM Siggraph Computer Graphics* 26.2, pp. 99–104.

Carnegie-Mellon Mocap Database (n.d.). http://http://mocap.cs.cmu.edu/.

Chen, Ching-Hang and Deva Ramanan (2017). "3D Human Pose Estimation= 2D Pose Estimation+ Matching". In: *CVPR*. IEEE, pp. 5759–5767.

Chentanez, Nuttapong et al. (2020). "Cloth and skin deformation with a triangle mesh based convolutional neural network". In: *Computer Graphics Forum*. Vol. 39. 8. Wiley Online Library, pp. 123–134.

De Aguiar, Edilson et al. (2010). "Stable spaces for real-time clothing". In: *ACM Transactions on Graphics (TOG)* 29.4, pp. 1–9.

Defferrard, Michaël, Xavier Bresson, and Pierre Vandergheynst (2016). "Convolutional neural networks on graphs with fast localized spectral filtering". In: *Advances in neural information processing systems*, pp. 3844–3852.

Dong, Qi, Shaogang Gong, and Xiatian Zhu (2017). "Multi-task curriculum transfer deep learning of clothing attributes". In: *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE, pp. 520–529.

Eichner, Marcin et al. (2012). "2d articulated human pose estimation and retrieval in (almost) unconstrained still images". In: *International journal of computer vision* 99.2, pp. 190–214.

Fang, Hao-Shu et al. (2018). "Learning pose grammar to encode human body configuration for 3d pose estimation". In: *Thirty-Second AAAI Conference on Artificial Intelligence*.

Feynman, Carl Richard (1986). "Modeling the appearance of cloth". PhD thesis. Massachusetts Institute of Technology.

Garland, Michael and Paul S Heckbert (1997). "Surface simplification using quadric error metrics". In: *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*. ACM Press/Addison-Wesley Publishing Co., pp. 209–216.

Gast, Theodore F et al. (2015). "Optimization integrator for large time steps". In: *IEEE transactions on visualization and computer graphics* 21.10, pp. 1103–1115.

Geng, Zhenglin, Daniel Johnson, and Ronald Fedkiw (2020). "Coercing machine learning to output physically accurate results". In: *Journal of Computational Physics* 406, p. 109099.

Guan, Peng et al. (2009). "Estimating human shape and pose from a single image". In: *ICCV*. IEEE, pp. 1381–1388.

Guan, Peng et al. (2012). "DRAPE: DRessing Any PErson." In: *ACM Trans. Graph.* 31.4, pp. 35–1.

Gundogdu, Erhan et al. (2019a). "Garnet: A Two-stream Network for Fast and Accurate 3D Cloth Draping". In: *IEEE International Conference on Computer Vision (ICCV)*. IEEE.

— (2019b). "Garnet: A two-stream network for fast and accurate 3d cloth draping". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 8739–8748.

Hahn, Fabian et al. (2014). "Subspace clothing simulation using adaptive bases". In: *ACM Transactions on Graphics (TOG)* 33.4, pp. 1–9.

Han, Xiaoguang, Chang Gao, and Yizhou Yu (2017). "DeepSketch2Face: a deep learning based sketching system for 3D face and caricature modeling". In: *ACM Transactions on graphics (TOG)* 36.4, pp. 1–12.

Haumann, David (1987). "Modeling the physical behavior of flexible objects". In: *Topics in Physically-based Modeling, Eds. Barr, Barrel, Haumann, Kass, Platt, Terzopoulos, and Witkin, SIGGRAPH Course Notes*.

Hoang, Van-Thanh and Kang-Hyun Jo (2018). "3D Human Pose Estimation Using Cascade of Multiple Neural Networks". In: *IEEE Transaction on Industrial Informatics*.

Holden, Daniel et al. (2019). "Subspace neural physics: Fast data-driven interactive simulation". In: *Proceedings of the 18th annual ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pp. 1–12.

Ionescu, Catalin et al. (2014). "Human3.6M: Large Scale Datasets and Predictive Methods for 3D Human Sensing in Natural Environments". In: *IEEE TPAMI* 36.7, pp. 1325–1339.

Jiang, Boyi et al. (2020). "BCNet: Learning Body and Cloth Shape from A Single Image". In: *arXiv preprint arXiv:2004.00214*.

Jin, Ning et al. (2020). "A Pixel-Based Framework for Data-Driven Clothing". In: *Computer Graphics Forum*. Vol. 39. 8. Wiley Online Library, pp. 135–144.

Kaldor, Jonathan M, Doug L James, and Steve Marschner (2008). "Simulating knitted cloth at the yarn level". In: *ACM SIGGRAPH 2008 papers*, pp. 1–9.

— (2010). "Efficient yarn-based cloth with adaptive contact linearization". In: *ACM SIGGRAPH 2010 papers*, pp. 1–10.

Kanazawa, Angjoo et al. (2018). "End-to-end recovery of human shape and pose". In: *CVPR*.

Kavan, Ladislav and Jiří Žára (2005). "Spherical blend skinning: a real-time deformation of articulated models". In: *Proceedings of the 2005 symposium on Interactive 3D graphics and games*, pp. 9–16.

Kavan, Ladislav et al. (2008). "Geometric skinning with approximate dual quaternion blending". In: *ACM Transactions on Graphics (TOG)* 27.4, pp. 1–23.

Kim, Doyub et al. (2013). "Near-exhaustive precomputation of secondary cloth effects". In: *ACM Transactions on Graphics (TOG)* 32.4, pp. 1–8.

Kolotouros, Nikos, Georgios Pavlakos, and Kostas Daniilidis (2019). "Convolutional Mesh Regression for Single-Image Human Shape Reconstruction". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4501–4510.

Lahner, Zorah, Daniel Cremers, and Tony Tung (2018). "Deepwrinkles: Accurate and realistic clothing modeling". In: *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 667–684.

Lassner, Christoph et al. (2017). "Unite the People: Closing the Loop Between 3D and 2D Human Representations". In: *IEEE CVPR*, pp. 4704–4713.

Le, Binh Huy and Zhigang Deng (2012). "Smooth skinning decomposition with rigid bones". In: *ACM Transactions on Graphics (TOG)* 31.6, pp. 1–10.

Lewis, John P, Matt Cordner, and Nickson Fong (2000). "Pose space deformation: a unified approach to shape interpolation and skeleton-driven deformation". In: *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pp. 165–172.

Lin, Kevin et al. (2015). "Rapid clothing retrieval via deep learning of binary codes and hierarchical search". In: *Proceedings of the 5th ACM on International Conference on Multimedia Retrieval*. ACM, pp. 499–502.

Liu, Tiantian, Sofien Bouaziz, and Ladislav Kavan (2017). "Quasi-newton methods for real-time simulation of hyperelastic materials". In: *ACM Transactions on Graphics (TOG)* 36.3, pp. 1–16.

Liu, Tiantian et al. (2013). "Fast simulation of mass-spring systems". In: *ACM Transactions on Graphics (TOG)* 32.6, pp. 1–7.

Loper, Matthew, Naureen Mahmood, and Michael J Black (2014). "MoSh: Motion and shape capture from sparse markers". In: *ACM Transactions on Graphics (TOG)* 33.6, p. 220.

Loper, Matthew et al. (2015a). "SMPL: A skinned multi-person linear model". In: *ACM Transactions on Graphics (TOG)* 34.6, p. 248.

Loper, Matthew et al. (Oct. 2015b). "SMPL: A Skinned Multi-Person Linear Model". In: *ACM Trans. Graphics (Proc. SIGGRAPH Asia)* 34.6, 248:1–248:16.

Luo, Chenxu, Xiao Chu, and Alan Yuille (2018). "OriNet: A Fully Convolutional Network for 3D Human Pose Estimation". In: *BMVC*.

Luvizon, Diogo C, David Picard, and Hedi Tabia (2018). "2d/3d pose estimation and action recognition using multitask deep learning". In: *CVPR*.

Ma, Qianli et al. (2019). "Dressing 3D Humans using a Conditional Mesh-VAE-GAN". In: *arXiv preprint arXiv:1907.13615*.

Macklin, Miles, Matthias Müller, and Nuttapong Chentanez (2016). "XPBD: position-based simulation of compliant constrained dynamics". In: *Proceedings of the 9th International Conference on Motion in Games*, pp. 49–54.

Madadi, Meysam, Hugo Bertiche, and Sergio Escalera (2020). "SMPLR: Deep learning based SMPL reverse for 3D human pose and shape recovery". In: *Pattern Recognition*, p. 107472.

— (2021a). "Deep unsupervised 3D human body reconstruction from a sparse set of landmarks". In: *International Journal of Computer Vision* 129.8, pp. 2499–2512.

Madadi, Meysam et al. (2021b). "Learning Cloth Dynamics: 3D+Texture Garment Reconstruction Benchmark". In: *Proceedings of the NeurIPS 2020 Competition and Demonstration Track*. Ed. by Hugo Jair Escalante and Katja Hofmann. Vol. 133. Proceedings of Machine Learning Research. PMLR, pp. 57–76. URL: `https://proceedings.mlr.press/v133/madadi21a.html`.

Magnenat-thalmann, Nadia et al. (1988). "Joint-Dependent Local Deformations for Hand Animation and Object Grasping". In: *In Proceedings on Graphics interface '88*, pp. 26–33.

Mahmood, Naureen et al. (2019). "AMASS: Archive of motion capture as surface shapes". In: *Proceedings of the IEEE International Conference on Computer Vision*, pp. 5442–5451.

Marcard, Timo von et al. (2018). "Recovering Accurate 3D Human Pose in The Wild Using IMUs and a Moving Camera". In: *European Conference on Computer Vision (ECCV)*.

Martin, Sebastian et al. (2011). "Example-based elastic materials". In: *ACM SIGGRAPH 2011 papers*, pp. 1–8.

Martinez, Julieta et al. (2017). "A simple yet effective baseline for 3d human pose estimation". In: *ICCV*.

Mehta, Dushyant et al. (2017). "Vnect: Real-time 3d human pose estimation with a single rgb camera". In: *ACM Transactions on Graphics (TOG)* 36.4, p. 44.

Moreno-Noguer, Francesc (2017). "3d human pose estimation from a single image via distance matrix regression". In: *CVPR*. IEEE, pp. 1561–1570.

Müller, Matthias et al. (2007). "Position based dynamics". In: *Journal of Visual Communication and Image Representation* 18.2, pp. 109–118.

Narain, Rahul, Armin Samii, and James F O'brien (2012). "Adaptive anisotropic remeshing for cloth simulation". In: *ACM transactions on graphics (TOG)* 31.6, pp. 1–10.

Newell, Alejandro, Kaiyu Yang, and Jia Deng (2016). "Stacked hourglass networks for human pose estimation". In: *ECCV*, pp. 483–499.

Nibali, Aiden et al. (2018). "3D Human Pose Estimation with 2D Marginal Heatmaps". In: *arXiv preprint arXiv:1806.01484*.

Niepert, Mathias, Mohamed Ahmed, and Konstantin Kutzkov (2016). "Learning convolutional neural networks for graphs". In: *International conference on machine learning*, pp. 2014–2023.

Nikolenko, Sergey I. (2019). "Synthetic Data for Deep Learning". In: *ArXiv* abs/1909.11512.

Omran, Mohamed et al. (2018). "Neural body fitting: Unifying deep learning and model based human pose and shape estimation". In: *3D Vision (3DV)*. IEEE, pp. 484–494.

Pan, Zherong, Hujun Bao, and Jin Huang (2015). "Subspace dynamic simulation using rotation-strain coordinates". In: *ACM Transactions on Graphics (TOG)* 34.6, pp. 1–12.

Patel, Chaitanya, Zhouyingcheng Liao, and Gerard Pons-Moll (2020). "Tailornet: Predicting clothing in 3d as a function of human pose, shape and garment style". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7365–7375.

Pavlakos, Georgios et al. (2017). "Coarse-to-fine volumetric prediction for single-image 3D human pose". In: *CVPR*. IEEE, pp. 1263–1272.

Pavlakos, Georgios et al. (2018). "Learning to Estimate 3D Human Pose and Shape from a Single Color Image". In: *CVPR*.

Pfaff, Tobias et al. (2014). "Adaptive tearing and cracking of thin sheets". In: *ACM Transactions on Graphics (TOG)* 33.4, pp. 1–9.

Pfaff, Tobias et al. (2020). "Learning mesh-based simulation with graph networks". In: *arXiv preprint arXiv:2010.03409*.

Pons-Moll, Gerard et al. (2017). "ClothCap: Seamless 4D clothing capture and retargeting". In: *ACM Transactions on Graphics (TOG)* 36.4, p. 73.

Provot, Xavier (1997). "Collision and self-collision handling in cloth model dedicated to design garments". In: *Computer Animation and Simulation'97*. Springer, pp. 177–189.

Provot, Xavier et al. (1995). "Deformation constraints in a mass-spring model to describe rigid cloth behaviour". In: *Graphics interface*. Canadian Information Processing Society, pp. 147–147.

Pumarola, Albert et al. (2019a). "3DPeople: Modeling the geometry of dressed humans". In: *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2242–2251.

Pumarola, Albert et al. (2019b). "Unsupervised Image-to-Video Clothing Transfer". In: *The IEEE International Conference on Computer Vision (ICCV) Workshops*.

Qi, Charles R et al. (2017). "Pointnet: Deep learning on point sets for 3d classification and segmentation". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 652–660.

Rahaman, Nasim et al. (2019). "On the spectral bias of neural networks". In: *International Conference on Machine Learning*. PMLR, pp. 5301–5310.

Richardson, Elad, Matan Sela, and Ron Kimmel (2016). "3D face reconstruction by learning from synthetic data". In: *2016 fourth international conference on 3D vision (3DV)*. IEEE, pp. 460–469.

Ros, German et al. (2016). "The SYNTHIA Dataset: A Large Collection of Synthetic Images for Semantic Segmentation of Urban Scenes". In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Santesteban, Igor, Miguel A Otaduy, and Dan Casas (2019). "Learning-Based Animation of Clothing for Virtual Try-On". In: *Computer Graphics Forum*. Vol. 38. 2. Wiley Online Library, pp. 355–366.

— (2022). "SNUG: Self-Supervised Neural Dynamic Garments". In: *arXiv preprint arXiv:2204.02219*.

Santesteban, Igor et al. (2021). "Self-Supervised Collision Handling via Generative 3D Garment Models for Virtual Try-On". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11763–11773.

Sarándi, Istvań et al. (2018). "Synthetic Occlusion Augmentation with Volumetric Heatmaps for the 2018 ECCV PoseTrack Challenge on 3D Human Pose Estimation". In: *ECCV PoseTrack Workshop*.

Shin, Dongjoe and Yu Chen (2019). "Deep Garment Image Matting for a Virtual Try-on System". In: *The IEEE International Conference on Computer Vision (ICCV) Workshops*.

Shoemake, Ken (1985). "Animating rotation with quaternion curves". In: *Proceedings of the 12th annual conference on Computer graphics and interactive techniques*, pp. 245–254.

Sigal, Leonid, Alexandru Balan, and Michael J Black (2008). "Combined discriminative and generative articulated pose and non-rigid shape estimation". In: *Advances in neural information processing systems*, pp. 1337–1344.

Sigal, Leonid, Alexandru O Balan, and Michael J Black (2010). "Humaneva: Synchronized video and motion capture dataset and baseline algorithm for evaluation of articulated human motion". In: *International journal of computer vision* 87.1-2, p. 4.

Sigal, Leonid et al. (2012). "Loose-limbed people: Estimating 3D human pose and motion using non-parametric belief propagation". In: *International journal of computer vision* 98.1, pp. 15–48.

Socher, Richard et al. (2012). "Convolutional-recursive deep learning for 3d object classification". In: *Advances in neural information processing systems*, pp. 656–664.

Stoll, Carsten et al. (2011). "Fast articulated motion tracking using a sums of gaussians body model". In: *ICCV*. IEEE, pp. 951–958.

Sun, Xiao et al. (2017). "Compositional human pose regression". In.

Sun, Xiao et al. (2018). "Integral human pose regression". In: *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 529–545.

Tan, Vince, Ignas Budvytis, and Roberto Cipolla (2017). "Indirect deep structured learning for 3D human body shape and pose prediction". In: *BMVC*.

Tang, Min et al. (2013). "A GPU-based streaming algorithm for high-resolution cloth simulation". In: *Computer Graphics Forum*. Vol. 32. 7. Wiley Online Library, pp. 21–30.

Tang, Min et al. (2018). "I-Cloth: Incremental collision handling for GPU-based interactive cloth simulation". In: *ACM Transactions on Graphics (TOG)* 37.6, pp. 1–10.

Teng, Yun, Miguel A Otaduy, and Theodore Kim (2014). "Simulating articulated subspace self-contact". In: *ACM Transactions on Graphics (TOG)* 33.4, pp. 1–9.

Terzopoulos, Demetri et al. (1987). "Elastically deformable models". In: *Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, pp. 205–214.

Tiwari, Garvita et al. (2020). "Sizer: A dataset and model for parsing 3d clothing and learning size sensitive 3d clothing". In: *arXiv preprint arXiv:2007.11610*.

Tome, Denis, Christopher Russell, and Lourdes Agapito (2017). "Lifting from the deep: Convolutional 3d pose estimation from a single image". In: *CVPR 2017 Proceedings*, pp. 2500–2509.

Tung, Hsiao-Yu et al. (2017). "Self-supervised learning of motion capture". In: *Advances in Neural Information Processing Systems*, pp. 5236–5246.

Varol, Gül et al. (2017a). "Learning from Synthetic Humans". In: *CVPR*.

Varol, Gul et al. (2017b). "Learning from synthetic humans". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 109–117.

Varol, Gül et al. (2018). "BodyNet: Volumetric Inference of 3D Human Body Shapes". In: *ECCV*.

Vassilev, Tzvetomir, Bernhard Spanlang, and Yiorgos Chrysanthou (2001). "Fast cloth animation on walking avatars". In: *Computer Graphics Forum*. Vol. 20. 3. Wiley Online Library, pp. 260–267.

Vidaurre, Raquel et al. (2020). "Fully Convolutional Graph Neural Networks for Parametric Virtual Try-On". In: *Computer Graphics Forum*. Vol. 39. 8. Wiley Online Library, pp. 145–156.

Vincent, Pascal et al. (2010). "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion". In: *Journal of machine learning research* 11.Dec, pp. 3371–3408.

Wang, Robert Y, Kari Pulli, and Jovan Popović (2007). "Real-time enveloping with rotational regression". In: *ACM SIGGRAPH 2007 papers*, 73–es.

Wang, Tuanfeng Y et al. (2018). "Learning a shared shape space for multimodal garment design". In: *arXiv preprint arXiv:1806.11335*.

Wang, Tuanfeng Y et al. (2019). "Learning an intrinsic garment space for interactive authoring of garment animation". In: *ACM Transactions on Graphics (TOG)* 38.6, pp. 1–12.

Wang, Xiaohuan Corina and Cary Phillips (2002). "Multi-weight enveloping: least-squares approximation techniques for skin animation". In: *Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pp. 129–138.

Weil, Jerry (1986). "The synthesis of cloth objects". In: *ACM Siggraph Computer Graphics* 20.4, pp. 49–54.

Wu, Zonghan et al. (2019). "A comprehensive survey on graph neural networks". In: *arXiv preprint arXiv:1901.00596*.

Xu, Hongyi and Jernej Barbič (2016). "Pose-space subspace dynamics". In: *ACM Transactions on Graphics (TOG)* 35.4, pp. 1–14.

Yang, Jinlong et al. (2018a). "Analyzing clothing layer deformation statistics of 3d human motions". In: *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 237–253.

Yang, Wei et al. (2018b). "3d human pose estimation in the wild by adversarial learning". In: *CVPR*.

Yu, Tao et al. (2019). "SimulCap: Single-View Human Performance Capture with Cloth Simulation". In: *arXiv preprint arXiv:1903.06323*.

Yuan, Yu-Jie et al. (2019). "Mesh Variational Autoencoders with Edge Contraction Pooling". In: *arXiv preprint arXiv:1908.02507*.

Zanfir, Andrei et al. (2018). "Deep network for the integrated 3d sensing of multiple people in natural images". In: *Advances in Neural Information Processing Systems*, pp. 8410–8419.

Zeller, Cyril (2005). "Cloth simulation on the GPU". In: *ACM SIGGRAPH 2005 Sketches*, 39–es.

Zhang, Chao et al. (2017). "Detailed, accurate, human shape estimation from clothed 3D scan sequences". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4191–4200.

Zhang, Meng et al. (2020). "Deep Detail Enhancement for Any Garment". In: *arXiv e-prints*, arXiv–2008.

Zhou, Xingyi et al. (2017). "Towards 3D Human Pose Estimation in the Wild: a Weakly-supervised Approach". In: *ICCV*.

Zhou, Yi et al. (2019). "On the continuity of rotation representations in neural networks". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5745–5753.

Zurdo, Javier S, Juan P Brito, and Miguel A Otaduy (2012). "Animating wrinkles by example on non-skinned cloth". In: *IEEE Transactions on Visualization and Computer Graphics* 19.1, pp. 149–158.