ABSTRACT

Title of Dissertation: DYNAMICS-INSPIRED GARMENT RECONSTRUCTION AND SYNTHESIS FOR SIMULATION-BASED VIRTUAL TRY-ON

Junbang Liang
Doctor of Philosophy, 2021

Dissertation Directed by: Professor Dinesh Manocha
Professor Ming Lin (co-advisor)
Department of Computer Science

E-Commerce has been growing at a rapid pace in recent years. People are now more likely to shop online than going to physical stores. Digital try-on systems, as one alternative way to improve the user experience and popularize online garment shopping, has drawn attention of many researchers [1]. However, the technology is still far from being practical and easy-to-use to replace physical try-on, mostly due to the gap in modeling and in demonstrating garment-fitting between the digital and the real worlds. The estimation of the hidden parameters of the garments plays an important role in closing the gap, examples including accurate reconstruction of human shapes and sizes through consumer devices, faithful estimation of garment materials via learning and optimization, user-friendly recovery of dressed garments, and fast and realistic visual rendering of animated try-on results. Although previous methods have made some progresses on these under-constrained problems, learning-based approaches have shown tremendous potential in making notable impact. I

propose to address the key open research issues above by adopting machine learning and optimization techniques.

To accurately reconstruct human shapes and sizes, I propose a learning-based *shape-aware* human body mesh reconstruction for both pose and shape estimation that is supervised directly on shape parameters. To estimate garment materials from video, I design a differentiable cloth simulation algorithm that can optimize its input variables to fit the data, thereby inferring physical parameters from observations and reaching desired control goals. To take advantage of joint learning and optimization, I further propose a joint estimation framework targeting human body and apparels through a close-loop iterative optimization. By extracting temporal information of both the body and the garment, it can also recover fabric material(s) of a garment from one single RGB video. To render realistic try-on results in close to real-time speed, I design a time-domain parallelization algorithm that maximizes the overall performance acceleration in distributed systems with minimal communication overhead. I further propose a semi-supervised learning framework to directly predict fit-accurate cloth draping on a wide range of body shapes.

In summary, my proposed learning-based frameworks focus on improving the efficiency, scalability, and capability of cloth simulation, and enable accurate hidden parameter estimation by exploiting cloth simulation for supervised learning and gradient-based feedback control.

DYNAMICS-INSPIRED GARMENT RECONSTRUCTION AND SYNTHESIS FOR SIMULATION-BASED VIRTUAL TRY-ON

by

Junbang Liang

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2021

Advisory Committee:
Professor Dinesh Manocha, Chair
Professor Min Wu
Professor Ming C. Lin
Professor Soheil Feizi
Professor Tom Goldstein

Acknowledgments

I would like to thank my advisors, Prof. Dinesh Manocha and Prof. Ming Lin. I was led to the fascinating world of virtual reality and managed to contribute to the cutting-edge technology thanks to their guidance. I would also want to thank my committee members who provide a lot of useful comments and insights to me to improve the dissertation.

It is not possible to finish this journey without all the kind help and support from my all of my friends and all the GAMMA members. Last but not the least, I would like to thank my parents for encouraging me to pursue my dream and giving me support all the time.

Table of Contents

Acknowledgements ii

Table of Contents iii

List of Tables vii

List of Figures viii

Chapter 1: Introduction 1
- 1.1 Learning-Based Human Body and Garment Estimation . . . 3
- 1.2 Differentiable Simulation for Material Optimization . . . 4
- 1.3 Simulation-Based Virtual Try-On . . . 5
- 1.4 Thesis Statement . . . 6
- 1.5 Main Results . . . 7
  - 1.5.1 Shape-Aware Human Reconstruction Using Multi-View Images 7
  - 1.5.2 Differentiable Simulation for Material Optimization . . . 8
  - 1.5.3 Joint Estimation of Human and Garment from Video . . . 9
  - 1.5.4 Time-Domain Parallelization for Accelerating Cloth Simulation 10
  - 1.5.5 Dynamics-Inspired Garment Draping Prediction . . . 10
- 1.6 Outline of Dissertation . . . 11

Chapter 2: Shape-Aware Human Reconstruction Using Multi-View Images 13
- 2.1 Introduction . . . 13
- 2.2 Related Work . . . 16
  - 2.2.1 Human Body Pose and Shape Recovering . . . 16
  - 2.2.2 Learning-Based Pose/Shape Estimations . . . 19
  - 2.2.3 Use of Synthetic Dataset . . . 19
- 2.3 Overview . . . 20
- 2.4 Model Architecture . . . 21
  - 2.4.1 3D Body Representation . . . 23
  - 2.4.2 Scalable Multi-View Framework . . . 24
  - 2.4.3 Training and Inferring . . . 25
  - 2.4.4 Implementation Details . . . 26
- 2.5 Data Preparation . . . 27
  - 2.5.1 Parameter Space Sampling . . . 28
  - 2.5.2 Human Body Motion Synthesis . . . 28

2.5.3 Cloth Registration and Simulation . . . 29
2.5.4 Multi-View Rendering . . . 29
2.6 Results . . . 31
2.6.1 Ablation Study . . . 31
2.6.2 Comparisons with Multi-View Methods . . . 34
2.6.3 Real-World Evaluations . . . 35
2.6.4 Multi-View Input in Daily Life . . . 36
2.6.5 Extra Test Results . . . 37
2.6.6 Additional Results on Real-World Images . . . 38
2.6.7 Comparison on Human3.6M with Single-View Methods . . . 40
2.6.8 Results Without Training on Synthetic Data . . . 40
2.6.9 Detailed Errors on Real World Evaluation . . . 41
2.6.10 Evaluation on *3D People in the Wild*. . . . 42
2.6.11 Running Time . . . 43
2.7 Conclusion and Future Work . . . 43

Chapter 3: Differentiable Simulation for Material Optimization 45
3.1 Introduction . . . 45
3.2 Related Work . . . 47
3.3 Differentiable Cloth Simulation . . . 49
3.3.1 Cloth Simulation Basics . . . 50
3.3.2 Overview . . . 53
3.3.3 Derivatives of the Physics Solve . . . 54
3.3.4 Dynamic Collision Detection and Response . . . 55
3.3.5 Derivatives of the Collision Response . . . 56
3.3.6 Derivations of the Gradient Computation . . . 62
3.4 Experiments . . . 65
3.4.1 Ablation Study . . . 66
3.4.2 Material Estimation . . . 67
3.4.3 Motion Control . . . 71
3.4.4 Collision-rich Motion Control . . . 73
3.5 Conclusion . . . 73

Chapter 4: Joint Estimation of Human and Garment from Video 75
4.1 Introduction . . . 75
4.2 Related Work . . . 77
4.3 Method Overview . . . 80
4.4 Garment Auto-encoder . . . 82
4.4.1 Two-Level Encoder-Decoder Structure . . . 83
4.4.2 Representative Point Set Extraction . . . 85
4.4.3 Training Losses . . . 85
4.4.4 Recovery from Point Clouds to Garment Meshes . . . 86
4.5 Material Estimation . . . 87
4.5.1 Single Frame Closed-Loop Estimation . . . 87
4.5.2 Temporal Estimation for Garment Material . . . 90

4.6 Data Preparation and Training . . . 91
4.6.1 Training Details . . . 92
4.7 Experiments . . . 93
4.7.1 Quantitative Analysis . . . 93
4.7.2 Qualitative Results . . . 95
4.7.3 Lab Experiments and User Study . . . 98
4.7.4 Ablation Study . . . 100
4.7.5 Latent Code Interpolation . . . 104
4.7.6 Additional Qualitative Results . . . 105
4.7.7 Application: Virtual Try-On . . . 108
4.8 Conclusion . . . 109

Chapter 5: Time-Domain Parallelization for Accelerating Cloth Simulation 112
5.1 Introduction . . . 112
5.2 Related Work . . . 115
5.2.1 Cloth Simulation . . . 115
5.2.2 Time Parallel Time Integration Method . . . 116
5.2.3 Parallel Cloth Simulation . . . 117
5.2.4 Hierarchical Structures and Multi-level Methods . . . 117
5.2.5 Mesh Upsampling . . . 118
5.3 Overview . . . 119
5.3.1 Two-Level Mesh Hierarchy Representation . . . 120
5.4 Time Domain Parallelization . . . 121
5.4.1 Static Temporal Partitioning . . . 122
5.4.2 Adaptive Partitioning . . . 124
5.4.3 Analysis on Performance Scalability . . . 127
5.5 Smooth State Transitioning . . . 128
5.5.1 Iterative Detail Recovery . . . 129
5.5.2 Convergence and Continuity . . . 131
5.5.3 Proof of Convergence of Algorithm 3 . . . 131
5.5.4 Iteration Number Estimation . . . 134
5.5.5 Implementation Details . . . 136
5.5.6 State Inconsistency . . . 137
5.6 Results . . . 138
5.6.1 Parameter and Scenario Setting . . . 138
5.6.2 Performance . . . 140
5.6.3 Smoothness . . . 147
5.6.4 Memory and Render Latency . . . 147
5.6.5 Limitations . . . 148
5.7 Conclusion and Future Work . . . 149

Chapter 6: Dynamics-Inspired Garment Draping Prediction 150
6.1 Introduction . . . 150
6.2 Related Work . . . 152
6.3 Method . . . 155

6.3.1 Encoder . . . 156
6.3.2 GCN-Based Decoder . . . 157
6.3.3 Spectral Domain Decomposition . . . 158
6.3.4 Loss Functions . . . 160
6.4 Physics-Enforced Optimization . . . 162
6.5 Experiments . . . 165
6.5.1 Data Generation . . . 165
6.5.2 Ablation Study . . . 166
6.5.3 Optimization for Semi-Supervision . . . 168
6.5.4 Optimization for Graphic Print . . . 169
6.5.5 Quantitative Comparisons . . . 171
6.5.6 Qualitative Results . . . 172
6.5.7 Generalization to Different Garment Sizes . . . 173
6.6 Conclusion . . . 175

Chapter 7: Conclusion 176
7.1 Summary of Results . . . 176
7.2 Limitations . . . 179
7.3 Future Work . . . 181

List of Tables

2.1 Comparison results on Human3.6M using MPJPE . . . 32
2.2 Comparison results on MPI_INF_3DHP . . . 32
2.3 Comparison results on my synthetic dataset . . . 33
2.4 Comparison on Human3.6M . . . 35
2.5 Comparison results on tape-measured data . . . 35
2.6 Results on MPI_INF_3DHP, validation set . . . 39
2.7 Results on MPI_INF_3DHP, test set . . . 39
2.8 Results on Human3.6M . . . 41
2.9 Percentages of errors in common measurements . . . 42
2.10 Evaluation on an unseen dataset . . . 43

3.1 Statistics of the backward propagation . . . 67
3.2 Results on the material parameter estimation task . . . 70
3.3 Motion control results . . . 72

4.1 Comparison on material estimation . . . 94
4.2 Quantitative comparison . . . 95
4.3 Lab experiment results . . . 98
4.4 Ablation study for different parts . . . 100
4.5 Test errors on the Multi-Garment Net dataset . . . 104
4.6 Comparison with previous works . . . 108

5.1 Notations and definition of my method . . . 120
5.2 Results on a higher-resolution mesh . . . 139
5.3 Comparison between different partition schemes . . . 141
5.4 Results in the extreme case . . . 143
5.5 Comparison with GPU method . . . 144

6.1 Encoders ablation . . . 166
6.2 Decoders ablation . . . 167
6.3 Losses ablation . . . 168
6.4 Self-correcting pipeline ablation study . . . 169
6.5 Adaptation to new materials . . . 170
6.6 Comparison with TailorNet . . . 172
6.7 Comparison for models on different sizes . . . 174

List of Figures

2.1 The network structure . . . 22
2.2 Detailed network structure of the regression block . . . 26
2.3 Examples of rendered synthetic images . . . 30
2.4 Prediction results compared to HMR . . . 37
2.5 Results on images with varying pose and shape . . . 38
2.6 Results on real-world multi-view images . . . 38
2.7 My model trained without synthetic data. . . . 41

3.1 Impact of perturbation . . . 59
3.2 Example frame from the ablation study . . . 66
3.3 Example frame from the material estimation scene . . . 68
3.4 Example frame from the motion control experiment . . . 70
3.5 A motion control scene with more obstacles . . . 72

4.1 Overall network structure . . . 81
4.2 The network structure of the garment auto-encoder . . . 82
4.3 My estimation pipeline . . . 88
4.4 The network structure for body and garment estimation . . . 88
4.5 Qualitative comparison . . . 97
4.6 Material transfer between videos . . . 97
4.7 Qualitative results . . . 97
4.8 Qualitative comparison with a real-world video . . . 100
4.9 Sample test images for the comparisons . . . 101
4.10 Interpolation between different garments . . . 101
4.11 Material transfer examples . . . 102
4.12 Interpolation results . . . 106
4.13 Qualitative Results . . . 107
4.14 Virtual try-on example . . . 109
4.15 Training data examples . . . 109
4.16 User study examples . . . 110

5.1 Simulated 'Karate' animation using my method . . . 112
5.2 An overview of my method . . . 115
5.3 Adaptive partitioning Algorithm . . . 125
5.4 An example comparison of the meshes . . . 128
5.5 Performance scaling result (large time step) . . . 140

5.6 Results with increasing length of the simulation . . . 140
5.7 Performance scaling result (small time step) . . . 142
5.8 Small scale parallelization comparison . . . 145
5.9 Large scale parallelization comparison . . . 145
5.10 More simulation results . . . 146
5.11 Refining results . . . 146

6.1 My model learns how to drape garments . . . 150
6.2 Overall structure of my network . . . 155
6.3 Visualization of the eigen decomposition . . . 156
6.4 Reconstructions for different numbers of coefficients. . . . 159
6.5 The semi-supervised self-correcting training pipeline. . . . 164
6.6 Bodies at BMI percentiles 10, 30, 50, 70 and 90%. . . . 165
6.7 Qualitative examples of self-correcting optimization . . . 171
6.8 Qualitative comparison with TailorNet . . . 173
6.9 Qualitative comparison with previous work . . . 174

Chapter 1: Introduction

Digital try-on system, as one important part of E-Commerce, has the potential to become one of the revolutionary technologies that change people's lives. However, its development is limited by some practical constraints, such as accurate sizing of the body and vivid try-on demonstrations.

There are several reasons why customers still prefer physical try-on. First, consumers are unsure if what they buy online will fit their bodies well. Although there exist general sizing systems for individuals, its lack of consistency and standardization across different brands and garment materials can often make it difficult to sizing the clothes, especially for those with non-standard body shapes and proportion. Accurate estimation of human body shapes is the key to make digital try-on work. Second, the fabric material is usually one of the key considerations when shopping for clothes. Different fabric materials affect how the garments look and fit on a body, how customers would wear it, and whether or not they would buy it. However, the correspondences between the actual material and its digital representation are not well understood, not to mention that an accurate material estimation and digital cloning from the real-world examples is challenging.

Visual effects from the customers' view is as critical as other factors. There are

two common presentations of garments: 2D image-based and 3D mesh with photo-realistic rendering. They have different advantages and drawbacks, but both need a large garment database for support. While creating a 3D garment model takes considerable labor, 2D images often suffer from the lack of variation and it is much more difficult to make customized changes. In either case, the try-on system would need a user-friendly design and manipulation backend to meet the customer's needs. Last, but not least, a fast and vivid animation of the garments in motion, along with the body movement, can considerably improve the user experience. Although it is not as critical as other factors, realistic visual rendering could effectively reduce the perceptual gap between the real-world and the virtual garments for online shopping.

Although previous methods have made some progresses on these under-constrained problems, learning-based approaches have shown tremendous potential in making notable impact. I propose to address the key open research challenges above by adopting machine learning and optimization techniques. These include:

- Accurate reconstruction of human and garment through consumer devices;
- Faithful estimation of fabric materials via learning and optimization;
- User-friendly recovery of dressed garments;
- A real-time cloth simulation system for customized animation; and
- Fast and realistic visual rendering of animated try-on results.

1.1 Learning-Based Human Body and Garment Estimation

Human appearance reconstruction is one of the key techniques for building a vivid and interactive virtual world. It can be applied to create a virtual avatar for various applications, such as virtual try-on or teleconferencing. It can also be used during character prototyping in computer animation. Human body reconstruction, consisting of pose and shape estimation, has been widely studied in a variety of areas, including digital surveillance, computer animation, special effects, and virtual/augmented environments. Most of existing works [2, 3, 4, 5, 6] focus on human-body reconstruction and recent advances have made significant progress in this area. However, the problem itself is naturally ambiguous, given limited input and occlusion. Although applying a predefined prior can alleviate this ambiguity, it is still insufficient in several cases, especially when a part of the body is occluded by clothing, or when the pose direction is perpendicular to the camera viewing plane. For example, when the human is walking towards the camera, it can be difficult to distinguish the difference between a standing vs. walking pose using a direct front-view image, while a side-view image could be more informative of the posture.

Moreover, recovery of garment properties, especially for physical material estimation, has been under-explored due to the complexity and the diversity of cloth dynamics and coupled interaction with a human body. Estimating worn garment materials using RGB frames of a video is highly challenging. Image features are often sparse, containing many noisy signals regarding the fabric materials worn on the body. An effective way to amplify useful signals is to estimate garment geometry

from images as a by-product. However, this is already an open problem far from being solved due to several reasons. First, garments have highly dynamical geometry that is not easy to capture and model. Previous works on garment modeling [7, 8, 9] and estimation [10, 11, 12] often propose solutions on one single type of garment, mostly t-shirts. Although the methods are also applicable to other garments, lack of generalization in capturing different garment geometries present a considerable barrier for real-world applications: users can only choose one of few pre-trained garment types and are not able to import new ones easily. Second, accurate estimation is often hindered by camera projection and human body occlusion. For example, the human-body estimation network may disagree with the garment reconstruction network in skeleton orientation due to the projection ambiguity (*e.g.* an arm is posed forward vs. backward), resulting in prediction misalignment. Therefore, without a general garment representation and an accurate geometry estimation, it is very difficult to regress the fabric materials solely from garment image sequences.

1.2 Differentiable Simulation for Material Optimization

Differentiable physics simulation is a powerful family of techniques that applies gradient-based methods to learning and control of physical systems [13, 14, 15, 16, 17]. It can enable optimization for control, and can also be integrated into neural network frameworks for performing complex tasks. My work focuses on cloth simulation, which relates to applications in robotics, computer vision, and computer graphics [8, 18, 19, 20, 21, 22, 23]. My goal is to enable differentiable cloth

simulation, which can provide a unified approach to a variety of inverse problems for cloth.

Differentiable cloth simulation is challenging due to a number of factors, which include the high dimensionality of cloth (as compared for example to rigid bodies [13]) and the need to handle contacts and collision. For example, a simple 16×16 grid-based cloth mesh has 289 vertices, 867 variables, and 512 faces when triangulated. Typical resolutions for garments would be at least many thousands, if not millions, of vertices and faces. Previous work that tackled differentiable simulation with collisions set up a static linear solver to account for all constraints [13]. In my simple example with cloth, the number of pairwise constraints would be at least $289 \times 512 = 140K$ for vertex-face collisions alone, which renders existing methods impractical even for this simple system. Even if a dynamic solver is applied upon collision, solving a dense linear system with such high dimensionality makes the gradient computation infeasible.

1.3 Simulation-Based Virtual Try-On

Drape prediction systems fall mainly in two categories: physics-based simulation and learning-based garment generation. Significant progress has been achieved in visual simulation of cloth over the past decades [24, 25, 26, 27]. Numerous algorithms have been proposed that achieve high accuracy and robustness for various 3D graphics applications, though real-time simulation remains illusive for complex simulation scenarios. Given recent advances in manycore and cloud computing, par-

allel computing has emerged as a possible alternative to achieve the desired runtime performance. Parallelization is a popular, practical way to achieve performance improvement. With a highly scalable parallelization scheme, physics-based simulation can be accelerated by orders of magnitude, enabling fast user feedback in virtual try-on.

As an alternative, learning-based cloth draping is also one of the key components in virtual try-on systems. With the help of a well-trained draping network, virtual try-on can predict quickly and accurately how garments look and fit on a body. Previously mentioned cloth simulation typically is prohibitively slow, while image-based try-on [28, 29] does not provide fit-accurate information. Apart from its use to avoid the dressing room, fast garment draping can also be a critical component in interactive character prototyping for a wide range of applications, like teleconferencing, computer animations, special effects or computer games.

1.4 Thesis Statement

Dynamic constraints can be effectively enforced in human body estimation, garment material and geometry reconstruction, simulation acceleration, and draping prediction for virtual try-on systems, by coupling machine learning and optimization methods with cloth simulation.

To support this thesis statement, I present five novel algorithms:

1. A learning-based *shape-aware* human body mesh reconstruction for both pose and shape estimation,

2. A differentiable simulation algorithm for fabric material optimization,

3. A joint estimation framework for estimating human body and apparels through a close-loop iterative optimization,

4. A time-domain parallelization algorithm to accelerate the simulation performance in distributed systems, and

5. A dynamics-inspired learning framework to directly predict the cloth draping on a wide range of body shapes.

1.5 Main Results

This dissertation presents five methods to support each component of the thesis statement. I list the main results obtained within these methods below:

1.5.1 Shape-Aware Human Reconstruction Using Multi-View Images

In Chapter 2, I propose a scalable neural network framework to reconstruct the 3D mesh of a human body from multi-view images, in the subspace of the SMPL model [30]. Use of multi-view images can significantly reduce the projection ambiguity of the problem, increasing the reconstruction accuracy of the 3D human body under clothing. The key contributions of this work include:

- A learning-based *shape-aware* human body mesh reconstruction using SMPL parameters for both pose and shape estimation that is supervised directly on shape parameters.

- A scalable, end-to-end, multi-view multi-stage learning framework to account for the ambiguity of the 3D human body (geometry) reconstruction problem from 2D images, achieving improved estimation results.

- A large simulated dataset, including *clothed* human bodies and the corresponding ground-truth parameters, to enhance the reconstruction accuracy, especially in shape estimation, where no ground-truth or supervision is provided in the real-world dataset.

- Accurate *shape* recovery *under occlusion of garments* by (a) providing the corresponding supervision and (b) deepening the model using the multi-view framework.

1.5.2 Differentiable Simulation for Material Optimization

In Chapter 3, I propose a differentiable cloth simulator that can be embedded as a layer in deep neural networks (DNN) for estimating fabric material parameters. Differentiable simulation provides an effective, robust framework for modeling cloth dynamics, self-collisions, and contacts in DNN. Due to the high dimensionality of the dynamical system in modeling cloth, traditional gradient computation for collision response can become impractical. To address this problem, I propose to compute the gradient directly using QR decomposition of a much smaller matrix. The key contributions of this work include:

- A dynamic collision detection scheme to reduce collision dimensionality.

- A novel gradient computation method of collision response using implicit differentiation.

- An optimized backpropagation algorithm using QR decomposition.

1.5.3 Joint Estimation of Human and Garment from Video

In Chapter 4, I propose a network model that uses video input to jointly estimate the human body, the garment shape, as well as fabric materials of the garment dressed on a human. During the estimation, I use a closed-loop optimization structure to share information between tasks and feed the learned garment features for temporal estimation of garment material type. The key contributions include:

- The first end-to-end neural network that recovers fabric material(s) of a garment from one single RGB video;

- A two-level auto-encoder for learning the latent space of garments through multi-scale feature coupling;

- Joint estimation of human body and apparels through a close-loop iterative optimization;

- The first garment prediction model that can account for arbitrary topologies and uses a feedback loop to the body estimation for prediction consistency.

1.5.4 Time-Domain Parallelization for Accelerating Cloth Simulation

In Chapter 5, I propose a novel time-domain parallelization technique that makes use of the two-level mesh representation to resolve the time-dependency issue and develop a practical algorithm to smooth the state transition from the corresponding coarse to fine meshes. A load estimation and a load balancing technique used in online partitioning are also proposed to maximize the performance acceleration. The key contributions of this work include:

- A time-domain parallelization algorithm supporting *adaptive meshes* with minimal communication overhead;
- Load estimation and load balancing techniques that maximize the overall performance acceleration;
- A practical state transitioning algorithm between low- and high-resolution simulations to recover details and ensure the visual quality of the simulated sequences.

1.5.5 Dynamics-Inspired Garment Draping Prediction

In Chapter 6, I propose a novel learning framework for draping prediction that can incorporate arbitrary loss functions at runtime composed of three key components. First, a physics-inspired supervision on a novel neural network. Second, an unsupervised optimization process coupled to the physics of individual garments at runtime. Finally, self-correction of the network based on the samples optimized in

the previous stage. The key contributions of this work include:

- A semi-supervised framework that enables easy integration of constraints into the deep learning model.
- Introduction of novel loss functions that encode geometric, physical, design, and tailoring constraints.
- A novel encoder/decoder network that effectively captures global and local features from the input and dynamically aggregates neighborhood information.
- A new self-correcting method based on data augmentation that enables both more accurate predictions and reduced data preparation time.

1.6 Outline of Dissertation

The subsequent chapters of this dissertation are organized as follows.

Chapter 2 introduces a new shape-aware human body estimation method that uses multi-view input for accurate reconstruction.

Chapter 3 presents my differentiable cloth simulation method that can compute the gradients efficiently and achieve faster optimization convergence than gradient-free methods.

Chapter 4 demonstrates a novel joint learning model that simultaneously predict human body shapes, garment geometry, and its fabric material in an end-to-end network using temporal garment geometry features represented by latent codes.

Chapter 5 presents my time-domain parallelization algorithm for accelerated

cloth simulation.

Chapter 6 describes a semi-supervised learning framework that integrates multiple geometric and physics constraints into learning garment draping on various bodies.

Chapter 2: Shape-Aware Human Reconstruction Using Multi-View Images

2.1 Introduction

In order to accurately reconstruct and synthesize garments in virtual try-on systems, it is necessary to obtain a precise estimation of the 3D user body mesh first. Human body reconstruction, including both pose and shape estimation, is an essential building block for realistic virtual try-on systems. Despite its importance in a large number of applications, human body reconstruction still remains a challenging and popular topic of interest. While direct 3D body scanning can provide excellent and sufficiently accurate results, its adoption is somewhat limited by the required specialized hardware. I propose a practical method that can estimate body pose and shape directly from a small set of images (typically 3 to 4) taken at several different view angles, which can be adopted in many applications, such as Virtual Try-On. Compared to existing scanning-based reconstruction, my proposed approach is much easier to use. Compared to previous image-based estimation methods, my method offers a higher degree of accuracy in shape estimation when the input human body is not within a normal range of body-mass index (BMI) and/or when the body is

wearing loose clothing. Furthermore, my framework is flexible in the number of images used and this feature considerably extends its applicability.

In contrast to many existing methods, I use "multi-view" images as input, referring to photos taken of the same person with *similar* poses from different viewing angles. They can be taken using specialized multi-view cameras, but it is not necessary (Sec. 2.6.4). Single-view images often lack the necessary and complete information to infer the pose and shape of a human body, due to the nature of projection transformation. By obtaining information from multiple view angles, the ambiguity from projection can be considerably reduced, and the body shape under loose garments can also be more accurately reconstructed.

Previous work on pose and shape estimation of a human body (see Sec. 2.2) mostly rely on optimization. One of the most important metrics used in these methods is the difference between the original and the estimated silhouette. As a result, these methods cannot be directly applied to images where the human wears loose garments, e.g. long coat, evening gown. The key insight of my method is as follows. When estimating a person's shape, how the human body interacts with the cloth, *e.g.* how a t-shirt is shaped due to the push by the stomach or the chest, provides more information than the silhouette of the person. So image features, especially those on clothes, play an important role in the shape estimation. With recent advances in deep learning, it is widely believed that the deep Convolutional Neural Network (CNN) structure can effectively capture these subtle visual details as activation values. I propose a multi-view multi-stage network structure to effectively capture visual features on garments from different view angles to more accurately

infer pose and shape information.

Given a limited number of images, I incorporate prior knowledge about the human body shape to be reconstructed. Specifically, I propose to use the Skinned Multi-Person Linear (SMPL) model [30], which uses Principal Component Analysis (PCA) coefficients to represent human body shapes and poses. In order to train the model to accurately output the coefficients for the SMPL model, a sufficient amount of data containing ground-truth information is required. However, to the best of my knowledge, no such dataset exists to provide multiple views of a loosely clothed body with its ground-truth shape parameters (i.e. raw mesh). Previous learning-based methods do not address the shape (geometry) recovery problem [31] or only output one approximation close to the standard mean shape of the human body [32], which is insufficient when recovering human bodies with largely varying shapes. Taking advantage of physically-based simulation, I design a system pipeline to generate a large number of multi-view human motion sequences with different poses, shapes, and clothes. By training on the synthetic dataset with ground-truth shape data, my model is "shape-aware", as it captures the statistical correlation between visual features of garments and human body shapes. I demonstrate in the experiments that the neural network trained using additional simulation data can considerably enhance the accuracy of shape recovery.

To sum up, the key contributions of my work include:

- A learning-based *shape-aware* human body mesh reconstruction using SMPL parameters for both pose and shape estimation that is supervised directly on

shape parameters.

- A scalable, end-to-end, multi-view multi-stage learning framework to account for the ambiguity of the 3D human body (geometry) reconstruction problem from 2D images, achieving improved estimation results.

- A large simulated dataset, including *clothed* human bodies and the corresponding ground-truth parameters, to enhance the reconstruction accuracy, especially in shape estimation, where no ground-truth or supervision is provided in the real-world dataset.

- Accurate *shape* recovery *under occlusion of garments* by (a) providing the corresponding supervision and (b) deepening the model using the multi-view framework.

2.2 Related Work

In this section, I survey recent works on human body pose and shape estimation, neural network techniques, and other related work that make use of synthetic data.

2.2.1 Human Body Pose and Shape Recovering

Human body recovery has gained substantial interest due to its importance in a large variety of applications, such as virtual environments, computer animation, and garment modeling. Previous works reduce the ambiguity from occlusion using

different assumptions and input data. They consist of four main categories: pose from images, pose and shape from images under tight clothing, scanned meshes, and images with loose clothing.

Pose From Images. Inferring 2D or 3D poses in images of one or more people is a popular topic in Computer Vision and has been extensively studied [33, 34, 35, 36, 37]. I refer to a recent work, VNect by Mehta *et al.* [31] that is able to identify human 3D poses from RGB images in real time using a CNN. By comparison, my method estimates the pose and shape parameters at the same time, recovering the entire human body mesh rather than only the skeleton.

Pose and Shape From Images under Tight Clothing. Previous work [38, 39, 40, 41, 42, 43] use the silhouette as the main feature or optimization function to recover the shape parameters. As a result, these methods can only be used when the person is wearing tight clothes, as shown in examples [44, 45]. By training on images with humans under various garments both in real and synthetic data, my method can learn to capture the underlying human pose and shape based on image features.

Pose and Shape From Scanned Meshes. One major challenge of recovering human body from scanned meshes is to remove the cloth mesh from the scanned human body wearing clothes [22]. Hasler *et al.* [46] used an iterative approach. They first apply a Laplacian deformation to the initial guess, before regularizing it based on a statistical human model. Wuhrer *et al.* [47] used landmarks of the scanned input throughout the key-frames of the sequences to optimize the body pose, while recovering the shape based on the 'interior distance' that helps constrain the mesh

to stay under the clothes, with temporal consistency from neighboring frames. Yang *et al*. [48] applies a landmark tracking algorithm to prevent excessive human labor. Zhang *et al*. [49] took more advantages of the temporal information to detect the skin and cloth region. As mentioned before, methods based on scanned meshes are limited: the scanning equipment is expensive and not commonly used. My method uses RGB images that are more common and thus much more widely applicable.

Pose and Shape from Images under Clothing. Bălan *et al*. [50] are the first to explicitly estimate pose and shape from images of clothed humans. They relaxed the loss on clothed regions and used a simple color-based skin detector as an optimization constraint. The performance of this method can be easily degraded when the skin detector is not helpful, *e.g*. when people have different skin colors or wear long sleeves. However, my method is trained on a large number of images, which does not require this constraint. Bogo *et al*. [51] used 2D pose machines to obtain joint positions and optimizes the pose and shape parameters based on joint differences and inter-penetration error. Lassner *et al*. [52] created a semi-automatic annotated dataset by incorporating a silhouette energy term on SMPLify [51]. They trained a Decision Forest to regress the parameter based on a much more dense landmark set provided by the SMPL model [30] during the optimization. Constraining the silhouette energy effect to a human body parameter subspace can reduce the negative impact from loose clothing, but their annotated data are from the optimization of SMPLify [51], which has introduced errors inherently. In contrast, I generate a large number of human body meshes wearing clothes, with the pose and shape ground-truth, which can then train the neural network to be "*shape-aware*".

2.2.2 Learning-Based Pose/Shape Estimations

Recently a number of methods have been proposed to improve the 3D pose estimation with calibrated multi-view input, either using LSTM [53, 54], auto-encoder [55, 56] or heat map refinement [57, 58]. They mainly focus on 3D joint positions without parameterization, thus not able to articulate and animate. Choy *et al*. [59] proposed an LSTM-based shape recovery network for general objects. Varol *et al*. [4] proposed a 2-step estimation on human pose and shape. However, both methods are largely limited by the resolution due to the voxel representation. In contrast, my method outputs the entire body mesh with parameterization, thus is articulated with a high-resolution mesh quality. Also, my method does not need the calibration of the camera, which is more applicable to in-the-wild images. Kanazawa *et al*. [32] used an iterative correction framework and regularized the model using a learned discriminator. Since they do not employ any supervision other than joint positions, the shape estimation can be inaccurate, especially, when the person is relatively over-weighted. In contrast, my model is more shape-aware due to the extra supervision from my synthetic dataset. Recent works [60, 61, 62] tackle the human body estimation problem using various approaches; my method offers better performance in either single- or multi-view inputs by comparison.

2.2.3 Use of Synthetic Dataset

Since it is often time- and labor-intensive to gather a dataset large enough for training a deep neural network, an increasing amount of attention is drawn

to synthetic dataset generation. Recent studies [21, 63] have shown that using a synthetic dataset, if sufficiently close to the real-world data, is helpful in training neural networks for real tasks. Varol *et al*. [64] built up a dataset (SURREAL) which contains human motion sequences with clothing using the SMPL model and CMU MoCap data [65]. While the SURREAL dataset is large enough and is very close to my needs, it is still insufficient in that (a) the clothing of the human is only a set of texture points on the body mesh, meaning that it is a tight clothing, (b) the body shape is drawn from the CAESAR dataset [66], where the uneven distribution of the shape parameters can serve as a "prior bias" to the neural network, and (c) the data only consists of single view images, which is not sufficient for my training. Different from [63, 64], my data generation pipeline is based on physical simulation rather than pasting textures on the human body, enabling the model to learn from more realistic images where the human is wearing looser garments. Recent works [67, 68] also generate synthetic data to assist training, but their datasets have only very limited variance on pose, shape, and textures to prevent from overfitting. In contrast, my dataset consists of a large variety of different poses, shapes, and clothing textures.

2.3 Overview

In this section, I give an overview of my approach. First, I define the problem formally. Then, I introduce the basic idea of my approach.

Problem Statement: Given a set of multi-view images, $\mathbf{I}_1 \ldots \mathbf{I}_n$, taken for the same person with the same pose, recover the underlying human body pose and

shape.

In the training phase, I set $n = 4$, *i.e.* by default I take four views of the person: front, back, left and right, although the precise viewing angles and their orders are not required, as shown in Sec. 2.4.3. To extend my framework to be compatible with single view images, I copy the input image four times as the input. For more detail about image ordering and extensions to other multi-view input, please refer to Sec. 2.4.3. I employ the widely-used SMPL model [30] as my mesh representation, for its ability to express various human bodies using low dimensional parametric structures.

As mentioned before, this problem suffers from ambiguity issues because of the occlusions and the camera projection. Directly training on one CNN as the regressor can easily lead to the model getting stuck in local minima, and it cannot be adapted to an arbitrary number of input images. Inspired by the residual network structure [69], I propose a multi-view multi-stage framework (Sec. 2.4) to address this problem. Since real-world datasets suffer from limited foreground/background textures and ground-truth pose and shape parameters, I make use of synthetic data as additional training samples (Sec. 2.5) so that the model can be trained to be more shape-aware.

2.4 Model Architecture

In this section, I describe the configuration of my network model. As shown in Fig. 2.1, I iteratively run my model for several stages of error correction. Inside

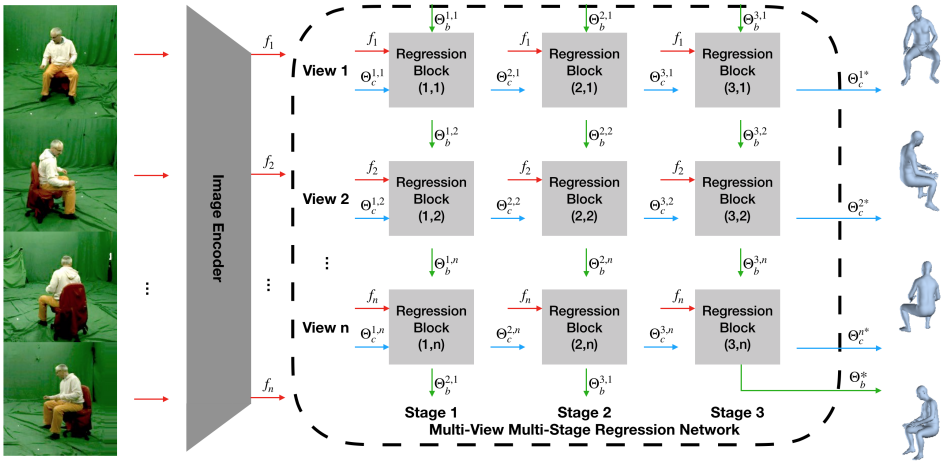


Figure 2.1: The network structure. Multi-view images are first passed through an image encoder to get feature vectors $f_1, ..., f_n$. With initial guesses of the camera parameters $\Theta_c^{1,i}$ and the human body parameters $\Theta_b^{1,1}$, the network starts to estimate the parameters stage by stage and view by view. Each regression block at the i^{th} stage and the j^{th} view regresses the corrective values from image feature f_j (red) and previous guesses $\Theta_c^{i,j}$ (blue) and $\Theta_b^{i,j}$ (green). The results will be added up to the input values and passed to future blocks. While the new human body parameters (green) can be passed to the next regression block, the view-specific camera parameters (blue) can only be passed to the next stage of the same view. Finally, the predictions of the n views in the last stage are outputted to generate the prediction.

each stage, the multi-view image input is passed on one at a time. At each step, the shared-parameter prediction block computes the correction based on the image feature and the input guesses. I estimate the camera and the human body parameters at the same time, projecting the predicted 3D joints back to 2D for loss computation. The estimated pose and shape parameters are shared among all views, while each view maintains its camera calibration and the global rotation. The loss at each step is the sum of the joint loss and the human body parameter loss:

$$L_i = \lambda_0 L_{2Djoint} + \lambda_1 L_{3Djoint} + L_{SMPL} \tag{2.1}$$

where λ_0 and λ_1 scale the units and control the importance of each term. I use L1 loss on 2D joints and L2 loss on others. L_{SMPL} is omitted if there is no ground-truth.

2.4.1 3D Body Representation

I use the Skinned Multi-Person Linear (SMPL) model [30] as my human body representation. It is a generative model trained from human mesh data. The pose parameters are the rotations of 23 joints inside the body, and the shape parameters are extracted from PCA. Given the pose and shape parameter, the SMPL model can then generate a human body mesh consisting of 6980 vertices:

$$\mathbf{X}(\theta, \beta) = \mathbf{W}\mathbf{G}(\theta)(\mathbf{X}_0 + \mathbf{S}\beta + \mathbf{P}\mathbf{R}(\theta)) \tag{2.2}$$

where $\mathbf{X} \in \mathbb{R}^{6980} \times \mathbb{R}^3$ is the computed vertices, $\theta \in \mathbb{R}^{72}$ are the rotations of each joint plus the global rotation, $\beta \in \mathbb{R}^{10}$ are the PCA coefficients, $\mathbf{W}, \mathbf{S}$ and $\mathbf{P}$ are trained matrices, $\mathbf{G}(\theta)$ is the global transformation, $\mathbf{X_0}$ are the mean body vertices, and $\mathbf{R}(\theta)$ is the relative rotation matrix.

For the camera model, I use orthogonal projection since it has very few parameters and is a close approximation to real-world cameras when the subject is sufficiently far away, which is mostly the case. I project the computed 3D body back to 2D for loss computation:

$$\mathbf{x} = s\mathbf{X}(\theta, \beta)\mathbf{R}^T + \mathbf{t} \tag{2.3}$$

where $\mathbf{R} \in \mathbb{R}^2 \times \mathbb{R}^3$ is the orthogonal projection matrix, s and $\mathbf{t}$ are the scale and the translation, respectively.

2.4.2 Scalable Multi-View Framework

My proposed framework uses a recurrent structure, making it a universal model applicable to the input of any number of views. At the same time, it couples the shareable information across different views so that the human body pose and shape can be optimized using image features from all views. As shown in Fig. 2.1, I use a multi-view multi-stage framework to couple multiple image inputs, with shared parameters across all regression blocks. Since the information from multiple views can interact with each other multiple times, the regression needs to run for several iterative stages. I choose to explicitly express this shared information as the predicted human body parameter since it is meaningful and also contains all of the information of the human body. Therefore the input of a regression block is the corresponding image feature vector and the predicted camera and human body parameters from the previous block. Inspired by the residual networks [69], I predict the corrective values instead of the updated parameters at each regression block to prevent gradient vanishing.

I have n blocks at each stage, where n is the number of views. Since all the input images contain the same human body with the same pose, these n blocks should output the same human-specific parameters but possibly different camera matrices. Thus I share the human parameter output across different views and the

camera transformation across different stages of the same view. More specifically, the regression block at the i^{th} stage and the j^{th} view takes an input of $(f_j, \Theta_c^{i,j}, \Theta_b^{i,j})$, and outputs the correction $\Delta\Theta_c^{i,j}, \Delta\Theta_b^{i,j}$, where f_j denotes the j^{th} image feature vector, $\Theta_c^{i,j}$ is the camera matrices and $\Theta_b^{i,j}$ is the human parameters. After that, I pass $\Theta_c^{i+1,j} = \Theta_c^{i,j} + \Delta\Theta_c^{i,j}$ to the next **stage** of the block at the same view, while I pass $\Theta_b^{i,j+1} = \Theta_b^{i,j} + \Delta\Theta_b^{i,j}$ to the next **block** of the chain (Fig. 2.1). At last, I compute the total loss as the average of the prediction of all n views in the final stage. Different from static multi-view CNNs which have to fix the number of inputs, I make use of the RNN-like structure in a cyclic form to accept any number of views, and avoid the gradient vanishing by using the error correction framework.

2.4.3 Training and Inferring

Intuitively I use $n = 4$ in my training process, since providing front, back, left, and right views can often give sufficient information about the human body. I choose a random starting view from the input images to account for the potential correlation between the first view and the initial guess. A specific order of the input views is not required since (a) the network parameters of each regression block are identical, and (b) none of the camera rotation information are shared among different views. To make use of large public single-view datasets, I copy each instance to 4 identical images as my input.

During inference, my framework can adapt to images with any number of views n as shown below. If $n \leq 4$, I use the same structure as used for training. I can pad

any of the input images to fill up the remaining views. As each view is independent in terms of global rotation, the choice of which view to pad does not matter. If $n > 4$, I extend my network to n views. Since this is an error-correction structure, the exceeded values introduced by extra steps can be corrected back. Note that the number of camera parameter corrections of each view always remains the same, which is the number of stages.

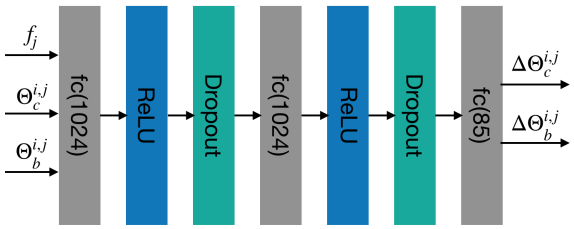


Figure 2.2: Detailed network structure of the regression block at the i^{th} stage and the j^{th} view. f_j denotes the image feature of the j^{th} view, $\Theta_c^{i,j}$ denotes the camera parameters, and $\Theta_b^{i,j}$ denotes the human body parameters.

2.4.4 Implementation Details

During training, besides my synthetic dataset for enhancing the shape estimation (detailed discussion in Sec. 2.5), I train on MS-COCO [70], MPI_INF_3DHP [71] and Human3.6M [72] datasets. Each mini-batch consists of half single view and half multi-view samples. Different from HMR [32], I do not use the discriminator. This is because (a) I initialized my parameters as the trained model of HMR [32], (b) the ground-truth given by my dataset serves as the regularization to prevent unnatural pose not captured by joint positions (*e.g.* foot orientations), and most importantly, (c) the ground-truth SMPL parameters from their training dataset does not have sufficient shape variety. Enforcing the discriminator to mean-shape biased dataset

will prevent the model to predict extreme shapes. I use 50-layer ResNet-v2 [73] for image feature extraction. The detailed structure inside the regression block is shown in Fig. 2.2. I fix the number of stages as 3 throughout the entire training and all testing experiments. The learning rate is set to 10^{-5}, and the training lasts for 20 epochs. Training on a GeForce GTX 1080 Ti GPU takes about one day.

2.5 Data Preparation

To the best of my knowledge, there is no public real-world dataset that captures motion sequences of human bodies, annotated with pose and shape (either using a parametric model or raw meshes), with considerable shape variation and loose garments. This lack of data, in turn, forces most of the previous human body estimations to focus only on joints. The most recent work [32] that recovers both pose and shape of human body does not impose an explicit shape-related loss function, so their model is not aware of varying human body shapes. In order to make my model shape-aware under clothing, I need data with ground-truth human body shapes where the garments should be dressed rather than pasted on the skin. A large amount of data is needed for training; sampling real-world data that captures the ground-truth shape parameters is both challenging and time-consuming. I choose an alternate method — using synthesized data. In this section, I propose an automatic pipeline to generate shape-aware training data, to enhance the shape estimation performance.

2.5.1 Parameter Space Sampling

I employ the SMPL model [30], which contains pose and shape parameters for human body. Pose parameters are rotation angles of joints. To sample meaningful human motion sequences in daily life, I use the CMU MoCap dataset [65] as my pose subspace. The shape parameters are principle component weights. It is not ideal to sample the shape parameters using Gaussian distribution; otherwise there will be many more mean-shape values than extreme ones, resulting in an unbalanced training data. To force the model to be more shape-aware, I choose to uniformly sample values at $[\mu - 3\sigma, \mu + 3\sigma]$ instead, where μ and σ represent the mean value and standard deviation of the shape parameters.

2.5.2 Human Body Motion Synthesis

After combining CMU MoCap pose data with the sampled shape parameters, it is likely that the human mesh generated by the SMPL model has inter-penetration due to the shape difference. Since inter-penetration is problematic for cloth simulation, I design an optimization scheme to avoid it in a geometric sense:

$$\min \|\mathbf{x} - \mathbf{x_0}\| \quad s.t. \quad g(\mathbf{x}) + \epsilon \leq 0 \tag{2.4}$$

where $\mathbf{x}$ and $\mathbf{x}_0$ stand for the vertex positions, $g(\mathbf{x})$ is the penetration depth, and ϵ is designed to reserve space for the garment. The main idea here is to avoid inter-penetrations by popping vertices out of the body, but at the same time keeping

the adjusted distance as small as possible, so that the body shape does not change much. This practical method works sufficiently well in most of the cases.

2.5.3 Cloth Registration and Simulation

Before I can start to simulate the cloth on each body generated, I first need to register them to the initial pose of the body. To account for the shape variance of different bodies, I first manually register the cloth to one of the body meshes. I mark the relative rigid transformation T of the cloth. For other body meshes, I compute and apply the global transformation, including both the transformation T and the scaling between two meshes. At last, I use the similar optimization scheme described in Sec. 2.5.2 to avoid any remaining collisions since it can be assumed that the amount of penetration after the transformation is small.

I use ArcSim [74] as the cloth simulator. I do not change the material parameters during the data generation. However, I do randomly sample the tightness of the cloth. I generally want both tight and loose garments in my training data.

2.5.4 Multi-View Rendering

I randomly apply different background and cloth textures in different sets of images. I keep the same cloth textures but apply different background across different views. I use the four most common views (front, back, left, and right), which are defined w.r.t. the initial human body orientation and fixed during the rendering. I sample 100 random shapes and randomly apply them to 5 pose sequences in

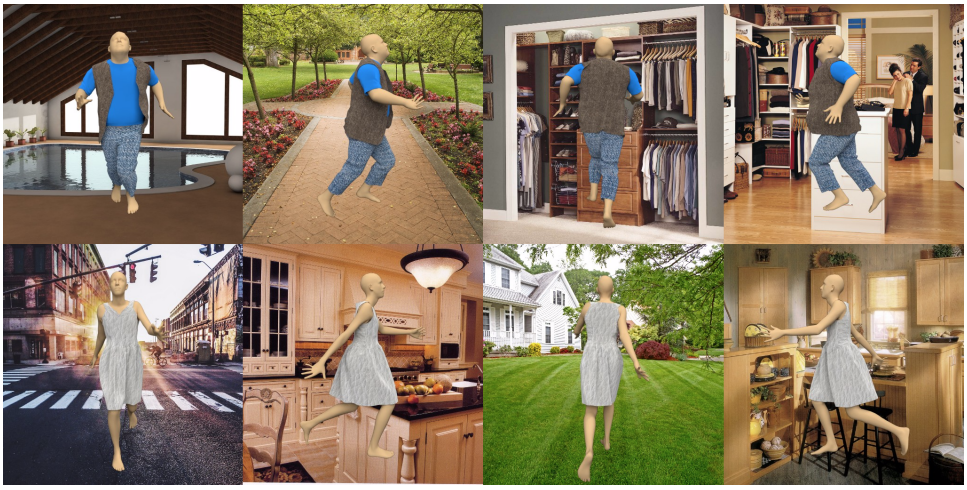


Figure 2.3: Examples of rendered synthetic images. I use a large number of real-world backgrounds and cloth textures so that the rendered images are realistic and diverse.

the CMU MoCap dataset (slow and fast walking, running, dancing, and jumping). After resolving collisions described in Sec. 2.5.3, I register two sets of clothes on it, one with a dress and the other with a t-shirt, pants, and jacket (Fig. 2.3). The pose and garment variety is arguably sufficient because (a) they provide most commonly seen poses and occlusions, and (b) it is an auxiliary dataset providing shape ground-truth which is jointly trained with real-world datasets that have richer pose ground-truth. I render two instances of each of the simulated frames, with randomly picked background and cloth textures. Given an average of 80 frames per sequence, I have generated 32,000 instances, with a total number of 128,000 images. I set the first 90 shapes as the training set and the last 10 as the test set. I ensure the generalizability across pose and clothing by coupling my dataset with other datasets with joint annotations (Sec. 2.4.4).

2.6 Results

I use the standard test set in Human3.6M and the validation set of MPI_INF_3DHP to show the performance gain by introducing multi-view input. Since no publicly available dataset has ground-truth shape parameters or mesh data, or data contains significantly different shapes from those within the normal range of BMI (*e.g.* over-weight or underweight bodies), I test my model against prior work (as the baseline) using the synthetic test set. Also, I test on real-world images to show that my model is more *shape-aware* than the baseline method – qualitatively using online images and quantitatively using photographs taken with hand-held cameras.

My method does not assume prior knowledge of the camera calibration so the prediction may have a scale difference compared to the ground-truth. There is also extra translation and rotation due to image cropping. To make a fair comparison against other methods, I report the metrics after a rigid alignment, following [32].

2.6.1 Ablation Study

I conduct an ablation study to show the effectiveness of my model and the synthetic dataset. In the experiments, HMR [32] is fine-tuned with the same learning setting.

2.6.1.1 Pose Estimation

I tested my model on datasets using multi-view images to demonstrate the strength of my framework. I use *Mean Per Joint Position Error* (MPJPE) of the

14 joints of the body, as well as *Percentage of Correct Keypoints* (PCK) at the threshold of 150mm along with *Area Under the Curve* (AUC) with threshold range 0-150mm [75] as my metrics. PCK gives the fraction of keypoints within an error threshold, while AUC computes the area under the PCK curve, presenting a more detailed accuracy within the threshold.

I use the validation set of MPI_INF_3DHP [32] as an additional test dataset since it provides multi-view input. It is not used for validation during my training. I also evaluated the original test set, which consists of single-view images.

Comparison: As shown in Table 2.1 and 2.2, under the same training condition, my model in single-view has similar, if not better, results in all experiments. Meanwhile, my model in multi-view achieves much higher accuracy.

Method	MPJPE w/ syn. training	MPJPE w/o syn. training
HMR	60.14	58.1
Mine (single)	58.55	59.09
Mine (multi)	**45.13**	**44.4**

Table 2.1: Comparison results on Human3.6M using MPJPE. Smaller errors implies higher accuracy.

Method	PCK/AUC/MPJPE w/ syn. training	PCK/AUC/MPJPE w/o syn. training
HMR	86/49/89	88/52/83
Mine (single)	88/52/84	87/52/85
Mine (multi)	**95/63/62**	**95/65/59**

Table 2.2: Comparison results on MPI_INF_3DHP in PCK/AUC/MPJPE. Better results have higher PCK/AUC and lower MPJPE.

2.6.1.2 Shape Estimation

To the best of my knowledge, there is no publicly available dataset that provides images with the captured human body mesh or other representation among a sufficiently diverse set of human shapes. Since most of the images-based datasets are designed for joint estimation, I decide to use my synthetic test dataset for large-scale statistical evaluation, and later compare with [32] using real-world images.

Other than MPJPE for joint accuracy, I use the Hausdorff distance between two meshes to capture the shape difference to the ground-truth. The Hausdorff distance is the maximum shortest distance of any point in a set to the other set, defined as follows:

$$d(V_1, V_2) = \max(\hat{d}(V_1, V_2), \hat{d}(V_2, V_1)) \tag{2.5}$$

$$\hat{d}(V_1, V_2) = \max_{\mathbf{u} \in V_1} \min_{\mathbf{v} \in V_2} \|\mathbf{u} - \mathbf{v}\|^2 \tag{2.6}$$

where V_1 and V_2 are the vertex set of two meshes in the same ground-truth pose, in order to negate the impact of different poses. Intuitively a Hausdorff distance of d means that by moving each vertex of one mesh by no more than d away, two meshes will be exactly the same.

Method	MPJPE/HD w/ syn. training	MPJPE/HD w/o syn. training
HMR	42/83	89/208
Mine (single)	44/65	102/283
Mine (multi)	**27**/**53**	**84**/273

Table 2.3: Comparison results on my synthetic dataset in MPJPE/Hausdorff Distance(HD). Better results have lower values.

As shown in Table 2.3, my model with multi-view input achieves the smallest error values, when compared to two other baselines. After joint-training with synthetic data, all models perform better in shape estimation, while maintaining similar results using other metrics (Table 2.1 and 2.2), i.e. they do not overfit. The joint errors of the HMR [32] are fairly good, so they can still recognize the synthesized human in the image. However, a larger Hausdorff distance indicates that they lose precision on the shape recovery.

Adding my synthetic datasets for training can effectively address this issue and thereby provide better shape estimation. I achieved a much smaller Hausdorff distance (with syn. training) even only using single view. This is because my refinement framework is effectively deeper, aiming at not only the pose but also the shape estimation, which is much more challenging than the pose-only estimation. With the same method, multi-view inputs can further improve the accuracy of shape recovery compared to results using only one single-view image.

2.6.2 Comparisons with Multi-View Methods

Since other multi-view methods only estimate human poses but not the entire body mesh, I compare the pose estimation results to them in Human3.6M. As shown in Table 2.4, I achieved state-of-the-art performance even when camera calibration is unknown and no temporal information is provided. As stated in Sec. 2.6, unknown camera parameters result in a scaling difference to the ground-truth, so the joint error would be worse than what it actually is. After the Procrustes alignment that

accounts for this effect, my method achieves the best MPJPE compared to other methods. Another potential source of the error is that my solution is constrained in a parametric subspace, while other methods output joint positions directly. In contrast, my method computes the entire human mesh in addition to joints and the result can be articulated and animated directly.

Method	MPJPE	Known Camera?	Run Time	Temporal Opt?	Articulated?	Shape?
Rhodin *et al.* [76]	-	Yes	0.025fps	Yes	No	Mix-Gaussian
Rhodin *et al.* [55]	98.2	Yes	-	Yes	No	No
Pavlakos *et al.* [57]	56.89	Yes	-	No	No	No
Trumble *et al.* [53]	87.3	Yes	25fps	Yes	No	No
Trumble *et al.* [56]	62.5	Yes	3.19fps	Yes	No	Volumetric
Núñez *et al.* [54]	54.21	Yes	8.33fps	Yes	No	No
Tome *et al.* [58]	52.8	Yes	-	No	No	No
Mine	79.85	No	33fps	No	Yes	Parametric
Mine (PA)	**45.13**					

Table 2.4: Comparison on Human3.6M with other multi-view methods. My method has comparable performance with previous work even without the assistance of camera calibration or temporal information. PA stands for Procrustes Aligned results.

2.6.3 Real-World Evaluations

Method	Standing	Sitting
HMR [32]	7.72%	7.29%
BodyNet [4]	13.72%	29.30%
Mine (single)	6.58%	10.18%
Mine (multi)	**6.23%**	**5.26%**

Table 2.5: Comparison results on tape-measured data using average relative errors (lower the better).

I first conduct a study on how my method performs differently with either single- or multi-view inputs under various conditions. My test subjects have two poses: standing and sitting, and the model is additionally tested on two sets of variants from the images. One is slightly dimmed, and the other has a large black

occlusion at the center of the first image. I use the percentage of errors from common body measurements used by tailors (*i.e.* lengths of neck, arm, leg, chest, waist, and hip), which is obtained using direct tape measurements on the subjects. I report the average relative error in Table 2.5. It is observed that single-view results are affected by the "occluded sitting" case, while the multi-view input can largely reduce the error. The reason why HMR is not impacted is that they uniformly output average human shapes for all input images. I also report results from BodyNet [4]. BodyNet outputs voxelized mesh and needs a time-consuming optimization to output the SMPL parameters. Its accuracy largely depends on the initial guess. Therefore, it resulted in a large amount of errors on the "sitting" case.

I also tested my model on other online images, where no such measurement can be done. As shown in Fig. 2.4, HMR [32] can predict the body pose but fails on inferring the person's shape. On the contrary, my model not only refines the relative leg orientations but also largely respects and recovers the original shape of the body.

2.6.4 Multi-View Input in Daily Life

It is often difficult to have multiple cameras from different view angles capturing a subject simultaneously. My model has the added benefit of not requiring the multi-view input be taken with the exact same pose. As the model has an error correction structure, it can be applied as long as the poses of the four views are not significantly different. I do not impose any assumptions on the background, so

(a) The input image. (b) My result. (c) HMR.

Figure 2.4: Prediction results compared to HMR. My model can better capture the shape of the human body. The recovered legs and chest are closer to the person in the image.

the images can be even taken with a fixed camera and a "rotating" human subject, which is the typically case when the method is used in applications like virtual try-on.

2.6.5 Extra Test Results

Table 2.6 and 2.8 shows the test results before Procrustes Alignment in MPI_INF_3DHP validation set and Human3.6M, respectively. The same conclusion about over-fitting and multi-view improvement can also be drawn from these data.

Table 2.7 shows the result in MPI_INF_3DHP test dataset. Since there is only one view fed into the model, the results are similar.

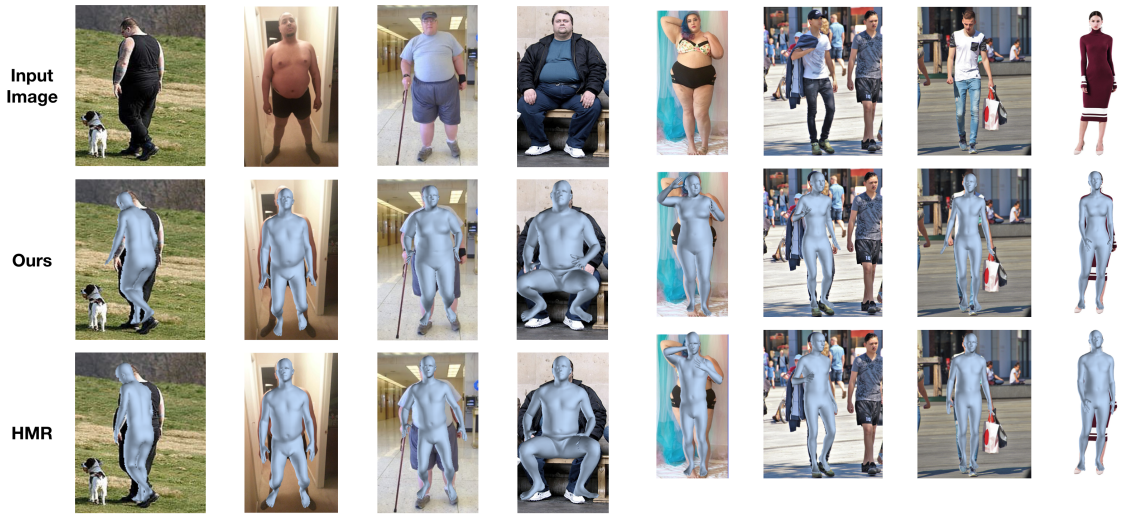


Figure 2.5: Results on images with varying pose and shape. The top row is the input image. The middle row shows my recovery results, and the bottom row shows the results from HMR [32]. Mine achieves better shape recovery results.

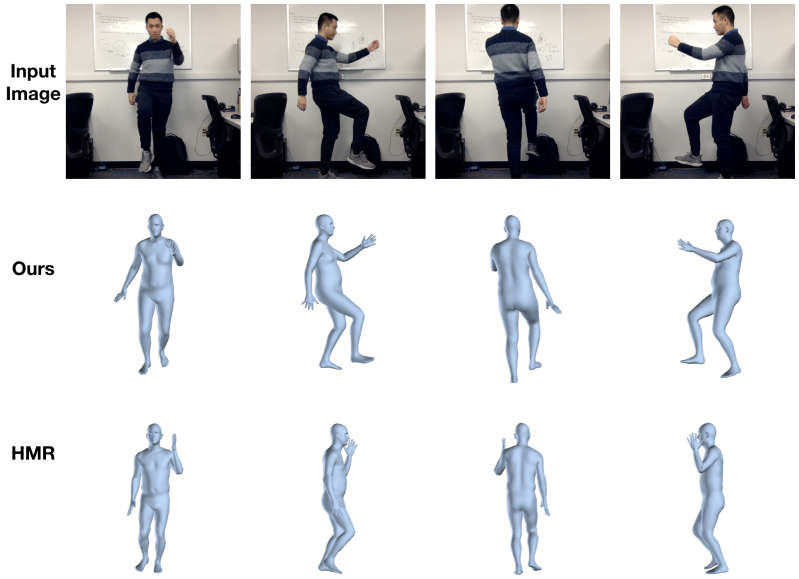


Figure 2.6: Results on real-world multi-view images. The top row is the input image. The middle row shows my recovery results, and the bottom row shows the results from HMR [32]. HMR is only given the front view as input. Mine achieves better pose recovery results due to more view angles.

2.6.6 Additional Results on Real-World Images

As shown in Fig. 2.5, given similar joint estimation results, my model captures more image features that indicate the shape of the human body and thereby gives

Method	PCK/AUC/MPJPE w/ syn. training	PCK/AUC/MPJPE w/o syn. training
HMR [32]	66/33/141	71/36/129
Mine (single)	69/32/139	68/33/138
Mine (multi)	72/34/128	72/35/126

Table 2.6: Results on MPI_INF_3DHP, validation set, before Procrustes aligment.

Method	PCK/AUC/MPJPE w/ syn. training	PCK/AUC/MPJPE w/o syn. training
HMR [32]	65/30/139	65/29/137
HMR (PA)	84/47/91	85/48/89
Mine	65/29/142	66/29/137
Mine (PA)	85/49/89	86/49/89

Table 2.7: Results on MPI_INF_3DHP, test set. The results of [32] are tested on cropped images by Mask-RCNN [77] so the values have minor difference than their reported ones. Only single view is available in this dataset.

much better results in terms of human shape. My method can distinguish between fat (Column 1-5) and slim (Column 6-8) persons, and between male and female. On the other hand, the output shapes from HMR are almost the same, which is around the mean shape value. By incorporating the shape-aware synthetic dataset, my method largely improves the recovery when the input human body does not have an average shape. I also tested with real-world multi-view images vs. single-view HMR. I feed the front view of the subject to HMR but input all views into my model. As shown in Fig. 2.6, the front view does not provide complete information of the subject pose, resulting in large pose errors on the limbs. By sharing information from more views (most importantly side views in this case), my model can effectively reduce the ambiguity from the camera projection and thereby provide good pose estimations across all views.

2.6.7 Comparison on Human3.6M with Single-View Methods

Table 2.8 shows the comparison with single-view results. As mentioned previously, the reason I don't have much better accuracy before rigid alignment is that:

- My method does not assume known camera, resulting in an unknown scaling difference to the real-world coordinates. After the Procrustes alignment, I achieved similar (and better with multi-view) performance.
- My solution is constrained in a subspace. Other methods output joint positions directly so they have more DOF and can be more accurate. However, my output is more comprehensive, as it contains the entire human mesh in addition to joints and the result can be articulated and animated directly.

Compared to Kolotouros *et al.* [62], my model is trained on a much more diverse dataset (*e.g.* MS-COCO), which means that the accuracy may not be minimized on the specific subset (Human 3.6M).

2.6.8 Results Without Training on Synthetic Data

I further tested another variant of my model, which is trained without synthetic data (Fig. 2.7). It achieves better joint estimation, but the recovered human body does not seem to be visually correct, especially at the end-effectors. This is because the joint-only supervision does not impose any constraints on the orientations of the end-effectors, resulting in an arbitrary guess. The HMR model [32] avoids this by adding a discriminator, which however could have negative impact on shape

Method	MPJPE	PA-MPJPE
Tome *et al.* [35]	88.39	-
Rogez *et al.* [78]	87.7	71.6
Mehta *et al.* [31]	80.5	-
Pavlakos *et al.* [33]	71.9	**51.23**
Mehta *et al.* [75]	68.6	-
Sun *et al.* [79]	**59.1**	-
Zhou *et al.* [37]	107.26	-
Debra *et al.* [80]	55.5	-
*Kolotouros *et al.* [62]	**74.7**	51.9
*Omran *et al.* [60]	-	59.9
*Pavlakos *et al.* [61]	-	75.9
*HMR [32]	87.97	58.1
*Mine (single-view)	88.34	58.55
*Mine (multi-view)	79.85	**45.13**

Table 2.8: Results on Human3.6M. My method results in smaller reconstruction errors compared to HMR [32]. * indicates methods that output both 3D joints *and* shapes.

estimations, as discussed in Sec. 2.4.4. My synthetic dataset provides a supervision to not only the joint positions but also the rotations, hence the model will learn a prior at the end-effectors, demonstrating more natural results.

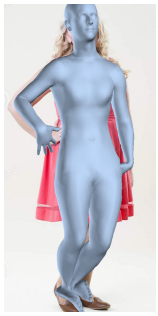


Figure 2.7: My model trained without synthetic data.

2.6.9 Detailed Errors on Real World Evaluation

The error percentages of each measure are shown in Table 2.9. Since the length of the arm and leg can be seen clearly in the front view, both inputs provide a

reasonably good estimation. However, given more views, my model can significantly reduce the error on other measurements, especially on those of chest, waist, and hip. I found that image illuminance has a negligible effect on the recovery result, which is due to the translation invariance of the convolutional layers. Occlusion has a notable impact on the recovery using only a single-view image, given only one view of the human body. However, by incorporating more views using my network model, the estimation can be considerably improved, indicating that the model using multi-view images is more robust to occlusion than with a single-view image as input.

error %	Regular				Dimmed				Partly Occluded			
input	Standing		Sitting		Standing		Sitting		Standing		Sitting	
# of views	Single	Multi	Single	Multi	Single	Multi	Single	Multi	Single	Multi	Single	Multi
neck	1.12	12.19	0.048	3.53	0.58	11.31	0.39	2.55	0.45	11.28	22.11	6.11
arm	4.76	4.22	8.03	7.33	6.21	4.95	8.10	6.89	5.20	3.82	7.20	6.70
leg	6.65	4.66	2.94	3.46	5.18	3.92	2.83	3.64	2.53	3.54	4.94	4.24
chest	4.59	7.72	8.40	3.1	6.20	7.20	8.13	3.19	19.80	1.57	30.04	13.72
waist	2.42	12.80	5.46	0.70	3.73	11.98	5.01	0.0084	13.78	8.52	30.05	10.61
hip	8.88	0.62	11.88	5.83	11.36	0.12	11.78	5.50	15.08	1.65	15.95	7.54
error %	Regular				Dimmed				Partly Occluded			
input	Standing		Sitting		Standing		Sitting		Standing		Sitting	
method	HMR	BodyNet	HMR	BodyNet	HMR	BodyNet	HMR	BodyNet	HMR	BodyNet	HMR	BodyNet
neck	10.4	2.9	4.8	26.3	8.4	1.6	4.6	26.2	9.2	3.9	5.7	6.8
arm	6.1	21.3	9.8	25.6	8.6	22.8	9.7	23.6	8.1	19.5	9.7	9.6
leg	7.9	6.3	1.8	4.4	4.3	6.6	1.8	3.3	5.1	6.2	2.1	3.0
chest	11.2	26.3	11.7	51.9	11.7	24.9	11.6	41.3	11.9	24.9	11.6	21.3
waist	9.4	9.0	8.7	42.7	9.4	7.7	8.5	33.7	9.7	8.3	8.4	11.4
hip	1.25	19.2	7.8	79.8	3.5	18.8	7.7	80	2.9	17	5.5	36.9

Table 2.9: Percentages of errors in common measurements of the human body under various lighting conditions using single-view vs. multi-view images. The multi-view model performs significantly better in estimating measurements of chest, waist, and hip, and is more robust, given variations in lighting and partial occlusion.

2.6.10 Evaluation on *3D People in the Wild*.

I have conducted the evaluation on *3D People in the Wild* dataset. As shown in Table 2.10, although the dataset consists of single view images of only a few subjects with nearly standard shapes, my model achieved better accuracy over HMR, while Alldieck *et al*. did not generalize well. The metric I used is mean joint error for

pose, and mean vertex error with ground-truth pose for shape.

2.6.11 Running Time

The previous work [32] trained 55 epochs for 5 days, while mine trained 20 epochs for 1 day. I list the training time here for reference, but it is actually not comparable since the batch size, epoch size and GPU type are not the same. In my environment, the inference time of HMR [32] is 2 microseconds while mine takes 7.5 (per view). This is because my network has a deeper structure to account for multiple views.

Method	Mean Joint Err.	Mean Vertex Err. (GT Pose)
HMR	93.77	21.71
Alldieck *et al.* [68]	169.61	47.07
Mine	96.86	20.96

Table 2.10: Evaluation on an unseen single-view dataset: *3D People in the Wild.* Values are mean joint error for pose and mean vertex error with ground-truth pose. My method has smaller errors than Alldieck *et al.*

2.7 Conclusion and Future Work

I proposed a novel multi-view multi-stage framework for pose and shape estimation. The framework is trained on datasets with at most 4 views but can be naturally extended to an arbitrary number of views. Moreover, I introduced a physically-based synthetic data generation pipeline to enrich the training data, which is very helpful for shape estimation and regularization of end effectors that traditional datasets do not capture. Experiments have shown that my trained model

can provide equally good pose estimation as state-of-the-art using single-view images, while providing considerable improvement on pose estimation using multi-view inputs and a better shape estimation across all datasets.

While synthetic data improves the diversity of human bodies with ground-truth parameters, a more convenient cloth design and registration are needed to minimize the performance gap between real-world images and synthetic data. In addition, other variables such as hair, skin color, and 3D backgrounds are subtle elements that can influence the perceived realism of the synthetic data at the higher expense of a more complex data generation pipeline. With the recent progress in image style transfer using GAN [81], a promising direction is to transfer the synthetic result to more realistic images to further improve the learning result.

This work has been published in the proceedings of the International Conference on Computer Vision (ICCV) 2019.

Chapter 3: Differentiable Simulation for Material Optimization

3.1 Introduction

Given the body estimation results in Chapter 2, the main task for building a virtual try-on system is to dress the given body with virtual garments. However, creating a cloth mesh that faithfully represents a real garment in a digital system is not easy. First, the geometry of the mesh needs to be designed well to fit the body. Second, the fabric materials of the designed garment are even more challenging to model correctly. While the geometry of the garment can still be intuitively hand crafted by experienced artists or designers, it is difficult to determine the fabric materials that exhibit mechanical properties closely resembling the real ones. There are two possible approaches to achieve this material cloning/estimation task. One way is to use supervised machine learning on manually labelled material data, which is labor-intensive and error-prone. The other method makes use of the geometric similarity instead of the direct material label, but needs a module that can compute the analytic relationship between the resulting geometry and the estimations or guesses of the fabric materials. Differentiable physics that can derive gradients regarding the simulation inputs can offer such a capability. With the differentiable simulation, one can either use an optimizer or train a network model to predict the

materials from the input, and compute its geometric appearance analytically. After comparing with the target shapes, a loss can then be created and back-propagate to the network, guiding the learning in a more intuitive way.

In this chapter, I propose a differentiable cloth simulation algorithm. First, I use dynamic collision detection since the actual collision pairs are very sparse. The collision response is solved by quadratic optimization, for which I can use implicit differentiation to compute the gradient. I directly solve the equations derived from implicit differentiation by using the QR decomposition of the constraint matrix, which is much smaller than the original linear system and is often of low rank. This approach reduces the gradient computation to a linear system of a small upper triangular matrix (the R component of the decomposition), which enables fast simulation and backpropagation.

My experiments indicate that the presented method makes differentiable cloth simulation practical. Using my method, the largest size of the linear system is 10x-20x smaller than the original solver in the backpropagation of the collision response, and the solver is 60x-130x faster. I demonstrate the potential of differentiable cloth simulation in a number of application scenarios, such as physical parameter estimation and motion control of cloth. With only a few samples, the differentiable simulator can optimize its input variables to fit the data, thereby inferring physical parameters from observations and reaching desired control goals.

3.2 Related Work

Differentiable physics. With recent advances in deep learning, there has been increasing interest in differentiable physics simulation, which can be combined with other learning methods to provide physically consistent predictions. Belbute-Peres *et al.* [13] and Degrave *et al.* [14] proposed rigid body simulators using a static formulation of the linear complementarity problem (LCP) [82, 83]. Toussaint *et al.* [15] developed a robot reasoning system that can achieve user-defined tasks and is based on differentiable primitives. Hu *et al.* [16] implemented a differentiable simulator for soft robots based on the Material Point Method (MPM). They store the object data at every simulation step so that the gradient can be computed out of the box. Schenck and Fox [17] embedded particle-based fluid dynamics into convolutional neural networks, with precomputed signed distance functions for collision handling. They solved or avoided collisions by assuming special object shapes, transferring to an Eulerian grid, or solving the corresponding collision constraint equation.

None of these methods can be applied to cloth simulation. First, cloth is a 2D surface in a 3D world; thus methods that use an Eulerian grid to compute material density, such as MPM [16], are not applicable. Second, the collision constraints in cloth simulation are more dynamic and complex given the high number of degrees of freedom; thus constructing a static dense LCP for the entire system [13, 14] or constructing the overall state transition graph [15] is inefficient and usually impossible for cloth of common resolution, since contact can happen for every edge-edge or vertex-face pair. Lastly, the shape of cloth changes constantly so self-collision

cannot be handled by precomputed signed distance functions [17].

In contrast, my method uses dynamic collision detection and computes the gradients of the collision response by performing implicit differentiation on the quadratic optimization used for computing the response. I utilize the low dimensionality and rank of the constraint matrix in the quadratic optimization and minimize the computation needed for the gradient propagation by giving an explicit solution to the linear system using QR decomposition of the constraint matrix.

Deep learning and physics. Supervised deep networks have been used to approximate physical dynamics. Mrowca *et al*. [84] and Li *et al*. [85] learned interaction networks to model particle systems. Ingraham *et al*. [86] trained a model to predict protein structures from sequences using a learnable simulator; the simulator predicts the deformation energy as an approximation to the physical process. Deep networks have also been used to support the simulation of fluid dynamics [87, 88, 89]. My method differs from many works that use deep networks to approximate physical systems in that I backpropagate through the true physical simulation. Thus my method conforms to physical law regardless of the scale of the problem. It can also naturally accept physical parameters as input, which enables learning from data.

Deep learning and cloth. Coupling cloth simulation with deep learning has become a popular way to solve problems such as detail refinement, garment retargeting, and material estimation. Yang *et al*. [21] proposed a recurrent model to estimate physical cloth parameters from video. Lähner *et al*. [23] trained a GAN to generate wrinkles on a coarse garment mesh which can then be automatically registered to a

human body using PCA. Gundogdu *et al.* [8] trained a graph convolutional framework to generate drapes and wrinkles given a roughly registered mesh. Santesteban *et al.* [90] developed an end-to-end retargeting network using a parametric human body model with displacements to represent the cloth.

These applications may benefit from my method. For garment retargeting problems, the relationship between body pose and vertex displacement is made explicit via the computed gradient, which can then be applied in network regularization for better performance. For parameter estimation, the differentiable simulation provides an optimization-based solution rather than a learning-based one. Instead of learning statistics from a large amount of data, I can directly apply gradient-based optimization via the simulator, which does not require any training data.

3.3 Differentiable Cloth Simulation

In this section, I introduce the main algorithms for the gradient computation. In general, I follow the computation flow of the common approach to cloth simulation: discretization using the finite element method [91], integration using implicit Euler [27], and collision response on impact zones [74, 92]. I use implicit differentiation in the linear solve and the optimization in order to compute the gradient with respect to the input parameters. The discontinuity introduced by the collision response is negligible because the discontinuous states constitute a zero-measure set. During the backpropagation in the optimization, the gradient values can be directly computed after QR decomposition of the constraint matrix.

3.3.1 Cloth Simulation Basics

Generally, cloth simulation includes three steps: force computation, dynamic solve, and collision handling. Extra steps, such as plasticity handling and strain limiting, are omitted since they are not essential components of a basic cloth simulation.

3.3.1.1 Force Computation

For external forces, the most common ones are gravity and wind forces, which are both straightforward. I focus on internal, constraint and frictional forces here.

Clothes are usually modeled as a 2D manifold mesh in 3D space. I apply Finite Element Method (FEM) to compute internal forces. For each triangle face in the mesh, I compute the deformation gradient as a variable of the strain:

$$\mathbf{F} = \frac{\partial \mathbf{x}}{\partial \mathbf{X}} \tag{3.1}$$

Here, $\mathbf{x}$ is the current 3D position of the triangle, and $\mathbf{X}$ is their coordinate in the 2D material space. Then, the stress (or internal forces) is computed using the deformation gradient $\mathbf{F}$. Usually a strain energy $\mathbf{E}$ is defined and I use its negative gradient as the force. In my base simulator, the stress is defined as a piece-wise linear function regarding the Green-Lagrange Strain, defined by Wang *et al.* [93]:

$$\mathbf{E} = \frac{1}{2}(\mathbf{F}^\top \mathbf{F} - \mathbf{I}) \tag{3.2}$$

Note that due to the geometric modeling of the cloth, there is no force caused by the thickness of the cloth. Most simulators use an extra 'bending force' as a compensation, following Grinspun *et al.* [94]. The bending force is defined between two adjacent faces when their dihedral angle is not a resting one.

The other two categories are relatively simpler. Constraint forces are defined as the negative gradient of the constraint energy, while frictional forces are created when two objects are in close proximity and have relative motions.

3.3.1.2 Dynamic Solve

In the simplest case, I solve $\mathbf{Ma} = \mathbf{f}$ for the acceleration and update the position and velocity accordingly, as shown in Algorithm 1. This Forward Euler method suffers from the well-known stability issue and often limits the time step size for the simulation. In order to take larger step for faster simulation, Backward Euler is often used. More specifically, I want my acceleration to match the force computed in the next time step:

$$\mathbf{M}\frac{\Delta \mathbf{v}}{\Delta t} = \mathbf{f}(\mathbf{x} + \Delta \mathbf{x}) = \mathbf{f}(\mathbf{x} + \Delta t(\mathbf{v} + \Delta \mathbf{v})) \tag{3.3}$$

By using Taylor Expansion, I have:

$$(\mathbf{M} - \Delta t^2 \frac{\partial \mathbf{f}}{\partial \mathbf{x}})\Delta \mathbf{v} = \Delta t \mathbf{f}(\mathbf{x} + \Delta t \mathbf{v}) \tag{3.4}$$

So the matrix used in the linear solve (Sec. 3.3.3) is defined as:

$$\hat{\mathbf{M}} = \mathbf{M} - \Delta t^2 \frac{\partial \mathbf{f}}{\partial \mathbf{x}} \tag{3.5}$$

As long as I have the Jacobian of the forces $\frac{\partial \mathbf{f}}{\partial \mathbf{x}}$, I can compute a more stable result of $\Delta \mathbf{v}$ and can apply larger Δt, as discussed by Baraff and Witkin [27].

3.3.1.3 Collision Handling

As introduced in Sec. 3.3.4, I used continuous collision detection between two simulation steps to detect all possible collisions. When two faces collide with each other, there are two different collision types: vertex-face collision and edge-edge collision. The common trait is that at time of collision, the four involved vertices are in the same plane. Based on this, I can develop and solve a cubic equation regarding the time of collision, t (Sec. 3.3.4).

When the collision is detected, I need to form the corresponding constraint at time t:

$$\left(\sum_{k=1}^{4} w_k \mathbf{x}_k(t) \right) \cdot \mathbf{n} \geq d \tag{3.6}$$

Here, w_k is the weight parameter, $\mathbf{x}_k(t)$ is the vertex position at time t, $\mathbf{n}$ is the normal of the plane, and d is the cloth thickness. The weight parameters are determined using barycentric coordinates of the intersection point in the face (in the vertex-face collision case) or on the edges (in the edge-edge collision case).

I consider w and $\mathbf{n}$ as constants during the optimization, and $\mathbf{x}_k(t)$ is linearly interpolated between two time steps. So it is a linear constraint regarding $\mathbf{x}$. Combining all constraints together, I have:

$$\mathbf{G}\mathbf{x} + \mathbf{h} \leq 0. \tag{3.7}$$

In the collision response phase, I want to introduce minimum energy to move the vertex away so that all constraints can be satisfied. Therefore, I form this optimization as a QP problem, as shown later in Sec. 3.3.5.

3.3.2 Overview

I begin by defining the problem formally and providing common notation. A triangular mesh $\mathcal{M} = \{\mathcal{V}, \mathcal{E}, \mathcal{F}\}$ consists of sets of vertex states, edges, and faces, where the state of the vertices includes both position $\mathbf{x}$ and velocity $\mathbf{v}$. Given a cloth mesh $\mathcal{M}_t$ together with obstacle meshes $\mathcal{M}_t^{obs}$ at step t, a cloth simulator can compute the mesh state $\mathcal{M}_{t+1}$ at the next step $t+1$ based on the computed internal and external forces and the collision response. A simple simulation pipeline is shown in Algorithm 1, where $\mathbf{M}$ is the mass matrix, $\mathbf{f}$ is the force, and $\mathbf{a}$ is the acceleration. For more detailed description of cloth simulation, please refer to Sec. 3.3.1. All gradients except the linear solve (Line 4 in Algorithm 1) and the collision response (Line 7) can be computed using automatic differentiation in PyTorch [95].

Algorithm 1: Cloth simulation

$\mathbf{v}_0 \leftarrow \mathbf{0}$
for $t = 1$ **to** n **do**
    $\mathbf{M}, \mathbf{f} \leftarrow$ compute_forces$(\mathbf{x}, \mathbf{v})$
    $\mathbf{a}_t \leftarrow \mathbf{M}^{-1}\mathbf{f}$
    $\mathbf{v}_t \leftarrow \mathbf{v}_{t-1} + \mathbf{a}_t \Delta t$
    $\mathbf{x}_t \leftarrow \mathbf{x}_{t-1} + \mathbf{v}_t \Delta t$
    $\mathbf{x}_t \leftarrow \mathbf{x}_t +$ collision_response$(\mathbf{x}_t, \mathbf{v}_t, \mathbf{x}_t^{obs}, \mathbf{v}_t^{obs})$
    $\mathbf{v}_t \leftarrow (\mathbf{x}_t - \mathbf{x}_{t-1})/\Delta t$
end for

3.3.3 Derivatives of the Physics Solve

In modern simulation algorithms, implicit Euler is often used for stable integration results. Thus the mass matrix $\mathbf{M}$ used in Algorithm 1 often includes the Jacobian of the forces. I denote it below as $\hat{\mathbf{M}}$ in order to mark the difference. A linear solve will be needed to compute the acceleration since it is time consuming to compute $\hat{\mathbf{M}}^{-1}$. I use implicit differentiation to compute the gradients of the linear solve. Given an equation $\hat{\mathbf{M}}\mathbf{a} = \mathbf{f}$ with a solution $\mathbf{z}$ and the propagated gradient $\frac{\partial \mathcal{L}}{\partial \mathbf{a}}|_{\mathbf{a}=\mathbf{z}}$, where $\mathcal{L}$ is the task-specific loss function, I can use the implicit differentiation form

$$\hat{\mathbf{M}}\partial\mathbf{a} = \partial\mathbf{f} - \partial\hat{\mathbf{M}}\mathbf{a} \tag{3.8}$$

to derive the gradient as

$$\frac{\partial \mathcal{L}}{\partial \hat{\mathbf{M}}} = -\mathbf{d}_\mathbf{a}\mathbf{z}^\top \quad \frac{\partial \mathcal{L}}{\partial \mathbf{f}} = \mathbf{d}_\mathbf{a}^\top, \tag{3.9}$$

where $\mathbf{d_a}$ is obtained from the linear system

$$\hat{\mathbf{M}}^\top \mathbf{d_a} = \frac{\partial \mathcal{L}}{\partial \mathbf{a}}^\top. \tag{3.10}$$

The proof is as follows. I take $\frac{\partial \mathcal{L}}{\partial \mathbf{f}}$ as an example here, the derivation of $\frac{\partial \mathcal{L}}{\partial \hat{\mathbf{M}}}$ is similar:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{f}} = \frac{\partial \mathcal{L}}{\partial \mathbf{a}} \cdot \frac{\partial \mathbf{a}}{\partial \mathbf{f}} = \mathbf{d_a}^\top \hat{\mathbf{M}} \cdot \hat{\mathbf{M}}^\dagger \mathbf{I} = \mathbf{d_a}^\top. \tag{3.11}$$

The first equality is given by the chain rule, the second is given by Equations 3.8 and 3.10, and $\hat{\mathbf{M}}^\dagger$ is the pseudoinverse of matrix $\hat{\mathbf{M}}$.

3.3.4 Dynamic Collision Detection and Response

As mentioned in Sec. 3.1, a static collision solver is not suitable for cloth because the total number of possible collision pairs is very high: quadratic in the number of faces. A common approach in cloth simulation is to dynamically detect collision on the fly and compute the response. I use a bounding volume hierarchy for collision detection [96], and non-rigid impact zones [92] to compute the collision response.

Specifically, I solve a cubic equation to detect the collision time t of each vertex-face or edge-edge pair that is sufficiently close to contact:

$$(\mathbf{x}_1 + \mathbf{v}_1 t) \cdot (\mathbf{x}_2 + \mathbf{v}_2 t) \times (\mathbf{x}_3 + \mathbf{v}_3 t) = 0, \tag{3.12}$$

where $\mathbf{x}_k$ and $\mathbf{v}_k$ $(k = 1, 2, 3)$ are the relative position and velocity to the first vertex. A solution that lies in $[0, 1]$ means that a collision is possible before the next simulation step. After making sure that the pair indeed intersects at time t, I set up one constraint for this collision, forcing the signed distance of this collision pair at time t to be no less than the thickness of the cloth δ. The signed distance of the vertex-face or edge-edge pair is linear to the vertex position $\mathbf{x}$. The set of all constraints then makes up a quadratic optimization problem as discussed later in Sec. 3.3.5.

For backpropagation, I need to compute the derivatives of the solution t since it is related to the parameters of the constraints. I use implicit differentiation here to simplify the process. Generally, given a cubic equation $ax^3 + bx^2 + cx + d = 0$, its implicit differentiation is of the following form:

$$(3ax^2 + 2bx + c)\partial x = \partial a x^3 + \partial b x^2 + \partial c x + \partial d. \tag{3.13}$$

Therefore I have

$$\begin{bmatrix} \frac{\partial x}{\partial a} & \frac{\partial x}{\partial b} & \frac{\partial x}{\partial c} & \frac{\partial x}{\partial d} \end{bmatrix} = \frac{1}{3ax^2 + 2bx + c} \begin{bmatrix} x^3 & x^2 & x & 1 \end{bmatrix}. \tag{3.14}$$

3.3.5 Derivatives of the Collision Response

A general approach to integrating collision constraints into physics simulation has been proposed by Belbute-Peres *et al.* [13]. However, as mentioned in Sections 3.1 and 3.2, constructing a static LCP is often impractical in cloth simulation

because of high dimensionality. Collisions that actually happen in each step are very sparse compared to the complete set. Therefore, I use a dynamic approach that incorporates collision detection and response.

Collision handling in my implementation is based on impact zone optimization [74]. It finds all colliding instances using continuous collision detection (Sec. 3.3.4) and sets up the constraints for all collisions. In order to introduce minimum change to the original mesh state, I develop a QP problem to solve for the constraints. Since the signed distance function is linear in $\mathbf{x}$, the optimization takes a quadratic form:

$$\underset{\mathbf{z}}{\text{minimize}} \quad \frac{1}{2}(\mathbf{z}-\mathbf{x})^\top \mathbf{W}(\mathbf{z}-\mathbf{x}) \tag{3.15}$$

$$\text{subject to} \quad \mathbf{G}\mathbf{z}+\mathbf{h} \leq \mathbf{0} \tag{3.16}$$

where $\mathbf{W}$ is a constant diagonal weight matrix related to the mass of each vertex, and $\mathbf{G}$ and $\mathbf{h}$ are constraint parameters. I further denote the number of variables and constraints by n and m, *i.e.* $\mathbf{x} \in \mathbb{R}^n$, $\mathbf{h} \in \mathbb{R}^m$, and $\mathbf{G} \in \mathbb{R}^{m \times n}$. Note that this optimization is a function with inputs $\mathbf{x}$, $\mathbf{G}$, and $\mathbf{h}$, and output $\mathbf{z}$. My goal here is to derive $\frac{\partial \mathcal{L}}{\partial \mathbf{x}}$, $\frac{\partial \mathcal{L}}{\partial \mathbf{G}}$, and $\frac{\partial \mathcal{L}}{\partial \mathbf{h}}$ given $\frac{\partial \mathcal{L}}{\partial \mathbf{z}}$, where $\mathcal{L}$ refers to the loss function.

When computing the gradient using implicit differentiation [97], the dimensionality of the linear system (Equation 3.20) can be too high. My key observation here is that $n \gg m > \text{rank}(\mathbf{G})$, since one contact often involves 4 vertices (thus 12 variables) and some contacts may be linearly dependent (*e.g.* multiple adjacent collision pairs). OptNet [97] solves a linear equation of size $m+n$, which is more than necessary. I introduce a simpler and more efficient algorithm below to minimize the

size of the linear equation.

3.3.5.1 QR Decomposition

To make things simpler, I assume that $\mathbf{G}$ is of full rank in this section. At global minimum $\mathbf{z}^*$ and λ^* of the Lagrangian, the following holds for stationarity and complementary slackness conditions:

$$\mathbf{W}\mathbf{z}^* - \mathbf{W}\mathbf{x} + \mathbf{G}^\top \lambda^* = 0 \tag{3.17}$$

$$D(\lambda^*)(\mathbf{G}\mathbf{z}^* + \mathbf{h}) = 0, \tag{3.18}$$

with their implicit differentiation as

$$\begin{bmatrix} \mathbf{W} & \mathbf{G}^\top \\ D(\lambda^*)\mathbf{G} & D(\mathbf{G}\mathbf{z}^* + \mathbf{h}) \end{bmatrix} \begin{bmatrix} \partial \mathbf{z} \\ \partial \lambda \end{bmatrix} = \begin{bmatrix} \mathbf{W}\partial \mathbf{x} - \partial \mathbf{G}^\top \lambda^* \\ -D(\lambda^*)(\partial \mathbf{G}\mathbf{z}^* + \partial \mathbf{h}) \end{bmatrix}, \tag{3.19}$$

where $D()$ transforms a vector to a diagonal matrix. Using similar derivation to Sec. 3.3.3, solving the equation

$$\begin{bmatrix} \mathbf{W} & \mathbf{G}^\top D(\lambda^*) \\ \mathbf{G} & D(\mathbf{G}\mathbf{z}^* + \mathbf{h}) \end{bmatrix} \begin{bmatrix} \mathbf{d}_\mathbf{z} \\ \mathbf{d}_\lambda \end{bmatrix} = \begin{bmatrix} \frac{\partial \mathcal{L}}{\partial \mathbf{z}}^\top \\ \mathbf{0} \end{bmatrix} \tag{3.20}$$

can provide the desired gradient:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{x}} = \mathbf{d}_{\mathbf{z}}^{T}\mathbf{W} \tag{3.21}$$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{G}} = -D(\lambda^{*})\mathbf{d}_{\lambda}\mathbf{z}^{*\top} - \lambda^{*}\mathbf{d}_{\mathbf{z}}^{\top} \tag{3.22}$$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{h}} = -\mathbf{d}_{\lambda}^{T}D(\lambda^{*}). \tag{3.23}$$

(See Sec. 3.3.6.2 for the derivation.) However, as mentioned before, directly solving Equation 3.20 may be computationally expensive in my case. I show that by performing a QR decomposition, the solution can be derived without solving a large system.

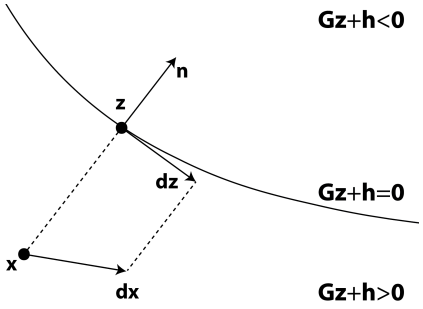


Figure 3.1: Impact of perturbation. A small perturbation of the target position will cause the final result to move along the constraint surface.

To further reduce computation, I assume that no constraint is 'over-satisfied', *i.e.* $\mathbf{G}\mathbf{z}^{*} + \mathbf{h} = \mathbf{0}$. I will remove these assumptions later in Sec. 3.3.5.2. I compute the QR decomposition of $\sqrt{\mathbf{W}}^{-1}\mathbf{G}^{\top}$:

$$\sqrt{\mathbf{W}}^{-1}\mathbf{G}^{\top} = \mathbf{Q}\mathbf{R}. \tag{3.24}$$

The solution of Equation 3.20 can be expressed as

$$\mathbf{d}_{\mathbf{z}} = \sqrt{\mathbf{W}}^{-1}(\mathbf{I} - \mathbf{Q}\mathbf{Q}^{\top})\sqrt{\mathbf{W}}^{-1}\frac{\partial \mathcal{L}}{\partial \mathbf{z}}^{\top} \tag{3.25}$$

$$\mathbf{d}_{\lambda} = D(\lambda^*)^{-1}\mathbf{R}^{-1}\mathbf{Q}^{\top}\sqrt{\mathbf{W}}^{-1}\frac{\partial \mathcal{L}}{\partial \mathbf{z}}^{\top}, \tag{3.26}$$

where $\sqrt{\mathbf{W}}^{-1}$ is the inverse of the square root of a diagonal matrix. The result above can be verified by substitution in Equation 3.20.

The intuition behind Equation 3.25 is as follows. When perturbing the original point $\mathbf{x}$ in an optimization, the resulting displacement of $\mathbf{z}$ will be moving along the surface of $\mathbf{G}\mathbf{x} + \mathbf{h} = \mathbf{0}$, which will become perpendicular to the normal when the perturbation is small. (Fig. 3.1 illustrates this idea in two dimensions.) This is where the term $\mathbf{I} - \mathbf{Q}\mathbf{Q}^{\top}$ comes from. Note that $\sqrt{\mathbf{W}}^{-1}\mathbf{G}^{\top}$ is an $n \times m$ matrix, where $n \gg m$ and the QR decomposition will only take $O(nm^2)$ time, compared to $O((n+m)^3)$ in the original dense linear solve. After that I will need to solve a linear system in Equation 3.26, but it is more efficient than solving Equation 3.20 since it is only of size m, and $\mathbf{R}$ is an upper-triangular matrix. In my collision response case, where $n \leq 12m$, my method can provide up to 183x acceleration in theory. The speed-up in my experiments (Sec. 3.4) ranges from 60x to 130x for large linear systems.

3.3.5.2 Low-rank Constraints

The algorithm above cannot be directly applied when $\mathbf{G}$ is low-rank, or when some constraint is not at boundary. This will cause $\mathbf{R}$ or $D(\lambda^*)$ to be singular. I now

show that the singularity can be avoided via small modifications to the algorithm.

First, if $\lambda_k = 0$ for the k^{th} constraint then $\mathbf{d}\lambda_k$ doesn't matter. This is because the final result contains only components of $D(\lambda^*)\mathbf{d}\lambda$ but not $\mathbf{d}\lambda$ alone, as shown in Equations 3.22 and 3.23. Intuitively, if the constraint is over-satisfied, then perturbing the parameters of that constraint will not have impact on $\mathbf{z}$. Based on this observation, I can remove the constraints in $\mathbf{G}$ when their corresponding λ is 0.

Next, if $\mathbf{G}$ is of rank k, where $k < m$, then I can rewrite Equation 3.24 as

$$\sqrt{\mathbf{W}}^{-1}\mathbf{G}^\top = \mathbf{Q}_1[\mathbf{R}_1 \quad \mathbf{R}_2], \tag{3.27}$$

where $\mathbf{Q}_1 \in \mathbb{R}^{n\times k}$, $\mathbf{R}_1 \in \mathbb{R}^{k\times k}$, and $\mathbf{R}_2 \in \mathbb{R}^{k\times(m-k)}$. Getting rid of $\mathbf{R}_2$ (*i.e.* removing those constraints from the beginning) does not affect the optimization result, but may change λ so that the computed gradients are incorrect. Therefore, I need to transfer the Lagrange multipliers to the linearly independent terms first:

$$\lambda_1 \leftarrow \lambda_1 + \mathbf{R}_1^{-1}\mathbf{R}_2\lambda_2, \tag{3.28}$$

where λ_1 and λ_2 are the Lagrange multipliers corresponding to the constraints on $\mathbf{R}_1$ and $\mathbf{R}_2$.

3.3.6 Derivations of the Gradient Computation

3.3.6.1 Proof of Equation 3.9

I now show that $\frac{\partial \mathcal{L}}{\partial \hat{\mathbf{M}}} = -\mathbf{d_a}\mathbf{z}^\top$. For convenience of expression, I split the matrix $\hat{\mathbf{M}}$ into elements $\{\hat{\mathbf{M}}_{i,j}\}$. Setting irrelevant variables to zero, I obtain from Equation 3.8 that:

$$\hat{\mathbf{M}}\partial\mathbf{a} = -\partial\hat{\mathbf{M}}\mathbf{z} = \begin{pmatrix} \mathbf{0} \\ -\partial\hat{\mathbf{M}}_{i,j}\mathbf{z}_j \\ \mathbf{0} \end{pmatrix} \tag{3.29}$$

Hence, I have:

$$\frac{\partial \mathbf{a}}{\partial \hat{\mathbf{M}}_{i,j}} = \hat{\mathbf{M}}^\dagger \begin{pmatrix} \mathbf{0} \\ -\mathbf{z}_j \\ \mathbf{0} \end{pmatrix} \tag{3.30}$$

Similar to Equation 3.11, I arrive at:

$$\frac{\partial \mathcal{L}}{\partial \hat{\mathbf{M}}_{i,j}} = \frac{\partial \mathcal{L}}{\partial \mathbf{a}} \cdot \frac{\partial \mathbf{a}}{\partial \hat{\mathbf{M}}_{i,j}} = \mathbf{d}_\mathbf{a}^\top \hat{\mathbf{M}} \cdot \hat{\mathbf{M}}^\dagger \begin{pmatrix} \mathbf{0} \\ -\mathbf{z}_j \\ \mathbf{0} \end{pmatrix} = -\mathbf{d}_{\mathbf{a}_i}\mathbf{z}_j \tag{3.31}$$

Combining all elements in $\hat{\mathbf{M}}$ together I have:

$$\frac{\partial \mathcal{L}}{\partial \hat{\mathbf{M}}} = -\mathbf{d_a}\mathbf{z}^\top \tag{3.32}$$

3.3.6.2 Proof of Equation 3.20-3.23

Let $\hat{\mathbf{z}} = \begin{bmatrix} \mathbf{z} & \lambda \end{bmatrix}^\top$. Using Equation 3.19 I have:

$$\frac{\partial \hat{\mathbf{z}}}{\partial \mathbf{x}} = \begin{bmatrix} \mathbf{W} & \mathbf{G}^\top \\ D(\lambda^*)\mathbf{G} & D(\mathbf{G}\mathbf{z}^* + \mathbf{h}) \end{bmatrix}^\dagger \begin{bmatrix} \mathbf{W} \\ \mathbf{0} \end{bmatrix} \tag{3.33}$$

$$\frac{\partial \hat{\mathbf{z}}}{\partial \mathbf{h}} = \begin{bmatrix} \mathbf{W} & \mathbf{G}^\top \\ D(\lambda^*)\mathbf{G} & D(\mathbf{G}\mathbf{z}^* + \mathbf{h}) \end{bmatrix}^\dagger \begin{bmatrix} \mathbf{0} \\ -D(\lambda^*) \end{bmatrix} \tag{3.34}$$

Then, the chain rule can yield the results as:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{x}} = \frac{\partial \mathcal{L}}{\partial \hat{\mathbf{z}}} \cdot \frac{\partial \hat{\mathbf{z}}}{\partial \mathbf{x}} \tag{3.35}$$

$$= \begin{bmatrix} \mathbf{d}_\mathbf{z}^\top & \mathbf{d}_\lambda^\top \end{bmatrix} \begin{bmatrix} \mathbf{W} & \mathbf{G}^\top \\ D(\lambda^*)\mathbf{G} & D(\mathbf{G}\mathbf{z}^* + \mathbf{h}) \end{bmatrix} \cdot \begin{bmatrix} \mathbf{W} & \mathbf{G}^\top \\ D(\lambda^*)\mathbf{G} & D(\mathbf{G}\mathbf{z}^* + \mathbf{h}) \end{bmatrix}^\dagger \begin{bmatrix} \mathbf{W} \\ \mathbf{0} \end{bmatrix} \tag{3.36}$$

$$= \mathbf{d}_\mathbf{z}^\top \mathbf{W} \tag{3.37}$$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{h}} = \frac{\partial \mathcal{L}}{\partial \hat{\mathbf{z}}} \cdot \frac{\partial \hat{\mathbf{z}}}{\partial \mathbf{h}} \tag{3.38}$$

$$= \begin{bmatrix} \mathbf{d}_{\mathbf{z}}^\top & \mathbf{d}_{\lambda}^\top \end{bmatrix} \begin{bmatrix} \mathbf{W} & \mathbf{G}^\top \\ D(\lambda^*)\mathbf{G} & D(\mathbf{G}\mathbf{z}^* + \mathbf{h}) \end{bmatrix} \cdot \begin{bmatrix} \mathbf{W} & \mathbf{G}^\top \\ D(\lambda^*)\mathbf{G} & D(\mathbf{G}\mathbf{z}^* + \mathbf{h}) \end{bmatrix}^\dagger \begin{bmatrix} \mathbf{0} \\ -D(\lambda^*) \end{bmatrix} \tag{3.39}$$

$$= -\mathbf{d}_{\lambda}^\top D(\lambda^*) \tag{3.40}$$

Similarly as Sec. 3.3.6.1, I split the matrix $\mathbf{G}$ into elements $\{\mathbf{G}_{i,j}\}$. From Equation 3.19 I have:

$$\begin{bmatrix} \mathbf{W} & \mathbf{G}^\top \\ D(\lambda^*)\mathbf{G} & D(\mathbf{G}\mathbf{z}^* + \mathbf{h}) \end{bmatrix} \partial \hat{\mathbf{z}} = \begin{bmatrix} -\partial \mathbf{G}^\top \lambda^* \\ -D(\lambda^*)\partial \mathbf{G}\mathbf{z}^* \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ -\partial \mathbf{G}_{i,j}\lambda_i^* \\ \mathbf{0} \\ -\lambda_i^* \partial \mathbf{G}_{i,j}\mathbf{z}_j^* \\ \mathbf{0} \end{bmatrix} \tag{3.41}$$

which indicates that:

$$\frac{\partial \hat{\mathbf{z}}}{\partial \mathbf{G}_{i,j}} = \begin{bmatrix} \mathbf{W} & \mathbf{G}^\top \\ D(\lambda^*)\mathbf{G} & D(\mathbf{G}\mathbf{z}^* + \mathbf{h}) \end{bmatrix}^\dagger \begin{bmatrix} \mathbf{0} \\ -\lambda_i^* \\ \mathbf{0} \\ -\lambda_i^* \mathbf{z}_j^* \\ \mathbf{0} \end{bmatrix} \tag{3.42}$$

So the chain rule gives:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{G}_{i,j}} = \frac{\partial \mathcal{L}}{\partial \hat{\mathbf{z}}} \cdot \frac{\partial \hat{\mathbf{z}}}{\partial \mathbf{G}_{i,j}} \tag{3.43}$$

$$= \begin{bmatrix} \mathbf{d}_{\mathbf{z}}^\top & \mathbf{d}_{\lambda}^\top \end{bmatrix} \begin{bmatrix} \mathbf{W} & \mathbf{G}^\top \\ D(\lambda^*)\mathbf{G} & D(\mathbf{G}\mathbf{z}^* + \mathbf{h}) \end{bmatrix} \cdot \begin{bmatrix} \mathbf{W} & \mathbf{G}^\top \\ D(\lambda^*)\mathbf{G} & D(\mathbf{G}\mathbf{z}^* + \mathbf{h}) \end{bmatrix}^\dagger \begin{bmatrix} \mathbf{0} \\ -\lambda_i^* \\ \mathbf{0} \\ -\lambda_i^* \mathbf{z}_j^* \\ \mathbf{0} \end{bmatrix} \tag{3.44}$$

$$= -\mathbf{d}_{\mathbf{z}_j} \lambda_i^* - \mathbf{d}_{\lambda_i} \lambda_i^* \mathbf{z}_j^* \tag{3.45}$$

Combining all elements in $\mathbf{G}$ together, I have:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{G}} = -\lambda^* \mathbf{d}_{\mathbf{z}}^\top - D(\lambda^*)\mathbf{d}_{\lambda}\mathbf{z}^{*\top} \tag{3.46}$$

3.4 Experiments

I conduct three experiments to showcase the power of differentiable cloth simulation. First, I use an ablation study to quantify the performance gained by using my method to compute the gradient. Next, I use the computed gradient to optimize the physical parameters of cloth. Lastly, I demonstrate the ability to control cloth motion.

3.4.1 Ablation Study

As mentioned in Sec. 3.3.5.1, my method for computing the gradients of the optimization can achieve a speed-up of up to 183x in theory. I conduct an ablation study to verify this estimate in practice. In order to clearly measure the timing difference, I design a scenario with many collisions. I put a piece of cloth into an upside-down square pyramid, so that the cloth is forced to fold, come into frequent contact with the pyramid, and collide with itself, as shown in Fig. 3.2.

I measure the running time of backpropagation in each quadratic optimization and also the running time of the physics solve as a reference. With all other variables fixed, I compare to the baseline method where the gradients are computed by directly solving Equation 3.20. Timings are listed in Tab. 3.1. In this experiment, the backpropagation of the physics solve takes from 0.007s to 0.5s, which, together with the timings of the baseline, implies that the collision handling step is the critical bottleneck when there are many collisions in the scene. The results in Tab. 3.1 show that my proposed method can significantly decrease the matrix size required for computation and thus the actual running time, resolving the bottleneck in backpropagation.

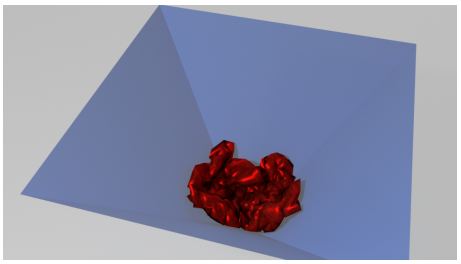


Figure 3.2: Example frame from the ablation study. A piece of cloth is crumpled inside a square pyramid, so as to generate a large number of collisions.

The experimental results also match well with the theory in Sec. 3.3.5. Each

collision involves a vertex-face or edge-edge pair, which both have 4 vertices and 12 variables. Therefore, the original matrix size ($n + m = 13m$) should be about 13 times bigger than in my method (m). In my experiment, the ratio of the matrix size is indeed close to 13. Possible reasons for the ratio not being exactly 13 include (a) multiple collision pairs that share the same vertex, making n smaller, and (b) the constraint matrix can be of low rank, as described in Sec. 3.3.5.2, making the effective m smaller in practice.

Mesh	Baseline		Mine		Speedup	
resolution	Matrix size	Runtime (s)	Matrix size	Runtime (s)	Matrix size	Runtime
16x16	599 ± 76	0.33 ± 0.13	$\mathbf{66 \pm 26}$	$\mathbf{0.013 \pm 0.0019}$	8.9	25
32x32	1326 ± 23	1.2 ± 0.10	$\mathbf{97 \pm 24}$	$\mathbf{0.011 \pm 0.0023}$	13	112
64x64	2024 ± 274	4.6 ± 0.33	$\mathbf{242 \pm 47}$	$\mathbf{0.072 \pm 0.011}$	8.3	64

Table 3.1: Statistics of the backward propagation with and without my method for various mesh resolutions. I report the average values in each cell with the corresponding standard deviations. By using my method, the runtime of gradient computation is reduced by up to two orders of magnitude.

3.4.2 Material Estimation

In this experiment, my aim is to learn the material parameters of cloth from observation. The scene features a piece of cloth hanging under gravity and subjected to a constant wind force, as shown in Fig. 3.3. I use the material model from Wang *et al.* [93]. It consists of three parts: density d, stretching stiffness $\mathbf{S}$, and bending stiffness $\mathbf{B}$. The stretching stiffness quantifies how large the reaction force will be when the cloth is stretched out; the bending stiffness models how easily the cloth can be bent and folded.

I used the real-world dataset from Wang *et al.* [93], which consists of 10 dif-

ferent cloth materials. There are in total 50 frames of simulated data. The first 25 frames are taken as input and all 50 frames are used to measure accuracy. This is a case-by-case optimization problem. My goal is to fit the observed data in each sequence as well as possible, with no "training set" used for training.

In my optimization setup, I use SGD with learning rate ranging from 0.01 to 0.1 and momentum from 0.9 to 0.99, depending on the convergence speed. The initial guess is the set of average values across all materials. I define the loss as the average MSE across all frames. In order to speed up optimization, I gradually increase the number of frames used. Specifically, I first optimize the parameters using only 1 simulated frame. I proceed to the second frame after the loss decreases to a certain threshold. This optimization scheme can help obtain a relatively good guess before additional frames are involved.

As a simple baseline, I measure the total external force and divide it by the observed acceleration to compute the density. For the stretching stiffness, I simplify the model to an isotropic one and record the maximum deformation magnitude along the vertical axis. Since the effect of the bending stiffness is too subtle to observe, I directly use the averaged value as my prior. I also compare my method with the L-BFGS optimization by Wang *et al*. [93] using finite difference. I used the PyTorch L-BFGS implementation and set the learning rate ranging from 0.1 to 0.2 depending on the convergence speed.

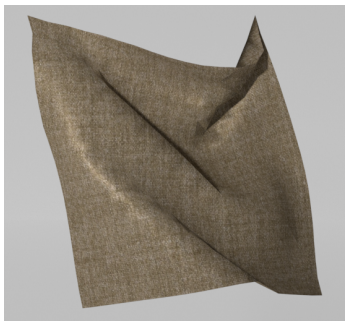


Figure 3.3: Example frame from the material estimation scene for cloth blowing in the wind.

For the performance measurement, I use the Frobenius norm normalized by the target as the metric for the material parameters:

$$\mathcal{E}(\mathbf{P}) = \frac{\|\mathbf{P} - \mathbf{P}_0\|_F}{\|\mathbf{P}_0\|_F}, \tag{3.47}$$

where $\mathbf{P}$ and $\mathbf{P}_0$ are the estimated and the target physics parameters, which stand for either density d, stretching stiffness $\mathbf{S}$, or bending stiffness $\mathbf{B}$. In order to show the final visual effect, I also measure the average distance of the vertices between the estimated one and the target normalized by the size of the cloth as another metric:

$$\mathcal{E}(\mathbf{X}) = \frac{1}{nTL} \sum_{1 \le i \le T, 1 \le j \le n} \|\mathbf{X}_{i,j} - \mathbf{Y}_{i,j}\|_2, \tag{3.48}$$

where L is the size of the cloth, and $\mathbf{X}$ and $\mathbf{Y}$ are $T \times n \times 3$ matrices denoting the n simulated vertex positions across T frames using the estimated parameter and the target, respectively.

Tab. 3.2 shows the estimation result. I achieve a much smaller error in most measurements in comparison to the baselines. The reason the stiffness matrices do not have low error is that (a) a large part of them describes the nonlinear stress behavior that needs a large deformation of the cloth and is not commonly observed in my environment, (b) different stiffness values can sometimes provide similar results, and (c) the bending force for common cloth materials is too small compared to gravity and the wind forces to make an impact. The table shows that the linear part of the stiffness matrix is optimized well. With the computed gradient using

my model, one can effectively optimize the unknown parameters that dominate the cloth movement to fit the observed data.

Compared with regular simulators, my simulator is designed to be embedded in deep networks. When gradients are needed, my simulator shows significant improvement over finite-difference methods, as shown in Tab. 3.2. Regular simulators need to run one simulation for each input variable to compute the gradient, while my method only needs to run once for all gradients to be computed. Therefore, the more input variables there are during learning, the greater the performance gain that can be achieved by my method over finite-difference methods.

Method	Runtime (sec/step/iter)	Density error (%)	Non-ln streching stiffness error (%)	Ln streching stiffness error (%)	Bending stiffness error (%)	Simulation error (%)
Baseline	-	68 ± 46	74 ± 23	160 ± 119	**70 ± 42**	12 ± 3.0
L-BFGS [93]	2.89 ± 0.02	4.2 ± 5.6	64 ± 34	72 ± 90	**70 ± 43**	4.9 ± 3.3
Mine	**2.03 ± 0.06**	**1.8 ± 2.0**	**57 ± 29**	**45 ± 41**	77 ± 36	**1.6 ± 1.4**

Table 3.2: Results on the material parameter estimation task. Lower is better. 'Ln' stands for 'linear'. Values of the material parameters are the Frobenius norms of the difference normalized by the Frobenius norm of the target. Values of the simulated result are the average pairwise vertex distance normalized by the size of the cloth. My gradient-based method yields much smaller error than the baselines.

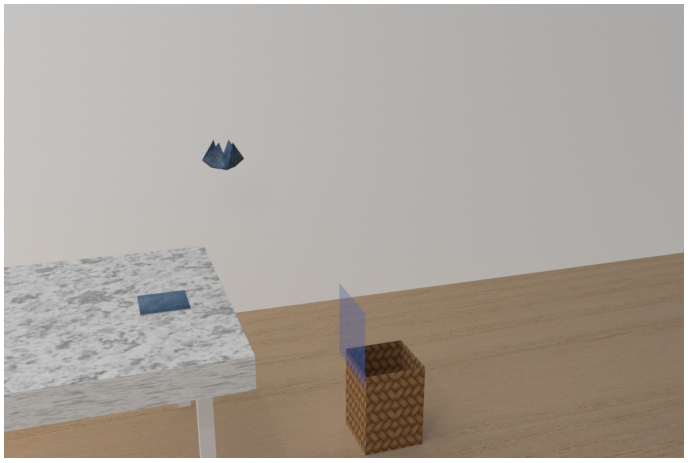


Figure 3.4: Example frame from the motion control experiment: dropping cloth into a basket.

3.4.3 Motion Control

I further demonstrate the power of my differentiable simulator by optimizing control parameters for motion control of cloth. The intended task is to drop a piece of cloth into a basket, as shown in Fig. 3.4. The cloth is originally placed on a table that is away from the basket. The system then applies external forces to the corners of the cloth to lift it and drop it into the basket. The external force is applied for 3 seconds and can be changed during this period. The basket is a box with an open top. A planar obstacle is placed between the cloth and the basket to increase the difficulty of the task.

The initial control force is set to zero. The control network consists of two FC layers, where the input (size $81 \times 2 \times 3$) is the position and velocity of each vertex, the hidden layer is of size 200, and the output is the control force (size 4×3). The learning rate is 10^{-4} and the momentum is 0.5. The reported result is the best among 10 trials.

I define the loss here as the squared distance between the center of mass of the cloth and the bottom of the basket. To demonstrate the ability to embed the simulator into neural networks, I also couple my simulator with a two-layer fully-connected (FC) network that takes the mesh states as input and outputs the control forces. My methods here are compared to two baselines. One of the baselines is a simple method that computes the momentum needed at every time step. The entire cloth is treated as a point mass and an external force is computed at each time step to control the point mass towards the goal. Obstacles are simply neglected

in this method. The other baseline is the PPO algorithm, as implemented in Ray RLlib [98]. The reward function is defined as the negative of the distance of the center of mass of the cloth to the bottom of the basket.

Method	Error (%)	Samples
Point mass	111	–
PPO [98]	432	10,000
Mine	**17**	**53**
Mine+FC	39	108

Table 3.3: Motion control results. The table reports the smallest distance to the target position, normalized by the size of the cloth, and the number of samples used during training.

Tab. 3.3 shows the performance of the different methods and their sample complexity. The error shown in the table is the distance defined above normalized by the size of the cloth. My method achieves the best performance with a much smaller number of simulation steps. The bottom of the basket in my setting has the same size as the cloth, so a normalized error of less than 50%, as my methods achieve, implies that the cloth is successfully dropped into the basket.

Figure 3.5: A motion control scene with more obstacles. The cloth needs to drop down and slide through the slopes to get to the target position.

3.4.4 Collision-rich Motion Control

I here demonstrate an example of motion control application with richer collisions. As shown in Figure 3.5, there is a series of obstacles above the basket that preclude the cloth from falling directly into it. The variable settings are the same as described in Sec. 3.4.3. My differentiable simulation provides the task with correct gradients so that the cloth is deposited into the basket.

3.5 Conclusion

I presented a differentiable cloth simulator that can compute the analytical gradient of the simulation function with respect to the input parameters. I used dynamic collision handling and explicitly derived its gradient. Implicit differentiation is used in computing gradients of the linear solver and collision response. Experiments have demonstrated that my method accelerates backpropagation by up to two orders of magnitude.

I have demonstrated the potential of differentiable cloth simulation in two application scenarios: material estimation and motion control. By making use of the gradients from the physically-aware simulation, my method can optimize the unknown parameters faster and more accurately than gradient-free baselines. Using differentiable simulation, I can learn the intrinsic properties of cloth from observation.

One limitation of my existing implementation is that the current simulation architecture is not optimized for large-scale vectorized operations, which introduces

some overhead. This can be addressed by a specialized, optimized simulation system based solely on tensor operations.

This work has been published in the conference proceedings of Neural Information Processing Systems (NeurIPS) 2019.

Chapter 4: Joint Estimation of Human and Garment from Video

4.1 Introduction

Chapter 2 and 3 introduce novel methods to estimate the human body and garment material independently. It is not difficult to observe that the two tasks are closely related. Jointly estimating both in an end-to-end system can help removing certain ambiguity originating from the input image. Moreover, estimating the garment materials of a worn garment is a more challenging task than what Chapter 3 introduces; it does not assume known external forces or even garment geometry, making differentiable physics not applicable under such a circumstance.

In this chapter, I introduce an end-to-end learning model that achieves both garment geometry estimation and fabric material prediction at the same time. To handle the dynamic geometry and different topologies of the garments and to provide a unified parametric model for the garments, I propose a two-level auto-encoder network. The key observation is that classical point cloud encoders such as PointNet [99] are great for capturing global shapes, but not suitable for encoding the local details. Multi-scale feature extraction not only decomposes the problem into smaller partitions, but also decouples global and local features to enable larger coverage on local shape learning. It is also critical to construct a continuous space that

includes different topological structures, since topology transitions often happen locally. During the estimation, I couple the human body inference with the garment recovery to maximize the estimation accuracy of the two correlated tasks. Other than traditional multi-tasking, I further introduce a closed-loop structure so that the garment features of different scales can guide the body estimation to improve the accuracy for both. Based on the temporal change of these garment features, I can then perform accurate material classification accordingly. To sum up, my key contributions include:

- The first end-to-end neural network that recovers fabric material(s) of a garment from one single RGB video (Section 4.3);
- A novel two-level auto-encoder for learning the latent space of garments through multi-scale feature coupling (Section 4.4);
- Joint estimation of human body and apparels through a close-loop iterative optimization (Section 4.5);
- A large dataset of garment motion sequences with wide variations of human body, fabric materials, textures, and lightings (Section 4.6);
- The first garment prediction model that can account for arbitrary topologies and use a feedback loop to the body estimation for prediction consistency (Section 4.7).

My experiments show that the proposed network structure effectively increases the capability and accuracy of the fabric material estimation. By using only a few frames

of a person wearing a garment, my model can faithfully reconstruct the garment fabric material(s), using the recovered shape and motion of both the garment and the human body as the conditioning in the material estimation task.

4.2 Related Work

Fabric material estimation. Researchers have been tackling different inverse problems, including inverse cloth design [100], combinatorial material design [101], BRDF parameter capturing [102], weaving pattern reconstruction [103], human material perception [104, 105], and frictional coefficient estimation [106, 107]. As a typical example of inverse problems, physical material estimation is a challenging yet important component of scene understanding. Cloth material estimation is even more challenging than most due to its highly dynamic motions. Previous works study the task in a simplified and constrained scenario, and recover the materials using statistical observation [108, 109], optimization [110, 111], or learning [21, 112]. In contrast, my method learns the cloth material from videos of a human wearing garments, which is more general and widely applicable. More importantly, my method makes use of the estimated multi-scale garment latent codes as input signal as well, which is shown to be much more useful in recovering overall garment geometry with local details than the traditional image features.

Garment modeling and estimation. Garment geometry capturing or recovery has been studied in computer graphics and computer vision. To address this problem, non-learning methods employ symmetry with user input [113], optimiza-

tion [110, 114], or binocular data [115]. Recently, methods using deep learning have been proposed for faster speed and more convenient usage [8, 10, 11, 12, 116, 117, 118, 119, 120, 121, 122]. In addition, direct garment modeling methods have also been proposed using spherical parameterization [7] for estimation or displacement map [123] for retargetting.

Different from displacement-based cloth representation [10], PCA-based models [9], and mesh-CNN-based methods [8], my garment model is universal to all topologies, applicable to those not homotopy to human surfaces (*e.g.* long dresses), and enabling semantic interpolation between different garments. Compared with [7], my model generates a stand-alone garment mesh, which is easy to export and retarget. More importantly, my method is the *first end-to-end network that jointly estimates the garment material together with its geometry.*

This method focuses on the *material* estimation of garments. Without any prior knowledge, I take an RGB video as input and recover the garment mesh, which is then used for fabric material estimation. My method is substantially different from most garment capturing or generation methods [90, 122, 124] regarding model input, output, and assumptions.

Point cloud encoder and decoder. PointNet [99] was among the first network model for encoding an unordered point set. Follow-on improvements include spatial partition [125, 126, 127], edge convolution [128], local region filtering [129, 130], and analogous convolutional operators [131].

Although these recent works have utilized hierarchical structure to some ex-

tent, their methods are not sufficient for auto-encoding the garment geometry or topology. The key difference between garment auto-encoding and rigid gadgets auto-encoding is that there are a large number of local details (*e.g.* wrinkles) due to cloth's highly deformable and dynamical nature. As a result, latent codes for local details are necessary. Methods using random sampling [131] or farthest point sampling [127] propose unstable representative points for similar input, thus not suitable for local feature encoding. [131] used a stabilized version of self-organized mapping (SOM) for representative point generation, but did not further encode the local features. So this method [131] is still a single-level auto-encoder.

In contrast, my two-level auto-encoder for garments successfully encodes local features, while capturing the global geometry. The trained latent code maintains similarity between similar input point clouds – otherwise not achievable using unstable representative points in prior works.

Human reconstruction from images. Human estimation using RGB images has been a popular research topic in deep learning for its importance in virtual reality and computer animation. While early works propose network models for only 2D/3D body skeletons [31, 132, 133], more recent works introduce techniques to regress the entire human body – either using a parametric human model [2, 3] or voxel-based representation [4, 5, 6]. Given the fact that the annotations in most real-world datasets contain only joint positions, the learning process has been refined in various ways [68, 134, 135, 136, 137].

In order to estimate the fabric material, I need to recover the garment shape

on the human body, which is an important problem rarely addressed in human estimation tasks. In my pipeline, I use state-of-the-art human body predictions as a strong prior for the garment estimation module. Given the focus of this paper, I use video input solely for garment material estimation. Thereby, I do not review related works on video-based pose estimation here. Instead, I refer the interested readers to a recent survey [138].

4.3 Method Overview

I first give the formal problem definition below. Given a video clip showing a person moving (*e.g.* walking, jumping, bending, etc), I estimate the fabric materials of the garment worn by the person. By *fabric materials*, I refer to the physical material parameters used in cloth simulation. I adopt the same material parameter definition introduced in [93], which consists of 24 parameters for stretching stiffness and 15 parameters for bending stiffness.

Due to the fact that the differences of the material parameter values do not intuitively reflect the human visual perception, I follow the previous work [21] to discretize the material parameter space based on the amount of deformations due to external forces. Using sensitivity analysis [139], the stretching stiffness is split into 6 classes and 9 for the bending. Combining both dimensions will yield 54 different material classes. As confirmed by [21], these 54 classes cover most of the common materials, including polyester, cotton, nylon, rayon, and their combinations. For example, one type of materials named *'white-swim-solid'* consisting of 87% nylon

and 13% spandex, as measured by [93], fits in the discrete classification model with the stretching label of 2 and the bending label of 3.

In this paper, I introduce an end-to-end deep neural network (Fig. 4.1) for simultaneously estimating the garment geometry and its material type(s), along with the human body. My key idea is that image features are not sufficient for inferring garment materials; it is necessary to extract the garment geometry as well for a more accurate estimation. To support different topologies of garments, I choose point clouds for its geometry representation. To better account for the highly dynamic garment surfaces, I train a two-level point cloud auto-encoder (Sec. 4.4) so that it can learn the global shapes and local features of the garment to reduce the total number of degrees of freedom. I use the SMPL model (see [30] for its rigorous math definition) to represent the human pose and shape.

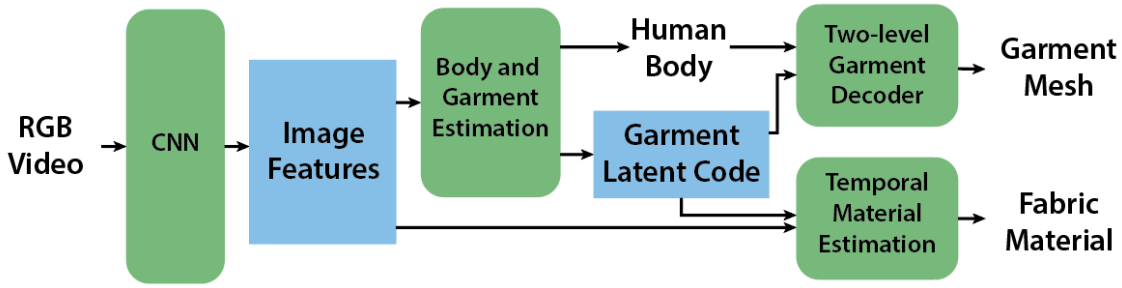


Figure 4.1: **Overall network structure**. Given an RGB video, I extract its image features and estimate the body and garment shape frame by frame (Sec. 4.5.1). The latter is decoded to obtain a garment mesh (Sec. 4.4). The temporal sequences of image and garment are fed to an LSTM for material classification (Sec. 4.5.2).

I divide the estimation pipeline into two phases. First, I estimate the human body and the cloth geometry in a frame-by-frame manner (Sec. 4.5.1). A closed-loop optimization structure is used to improve the estimation accuracy of these two correlated tasks. The garment geometry prediction module is conditioned on

the human body parameters, and at the same time provides corrective feedback to the human body prediction module. I then feed the features of the image and the garment geometry from each frame together to a temporal neural network for the garment material estimation (Sec. 4.5.2). By sharing common features, providing corrective feedback, and conditioning on outputs of closely-related tasks, my network model can achieve higher estimation accuracy on all three tasks than independent estimation baselines.

4.4 Garment Auto-encoder

I first set up an auto-encoder for the cloth model. Since the model is designed

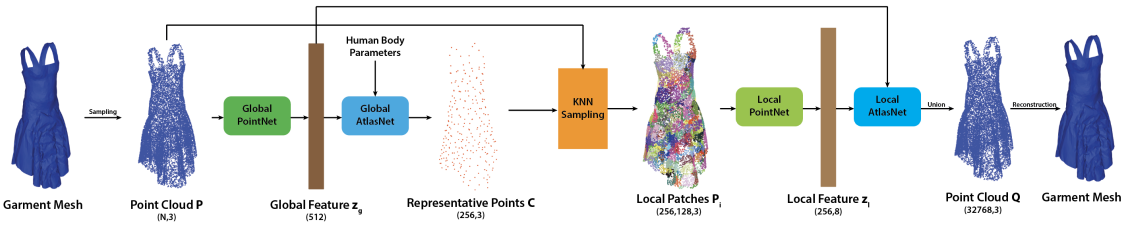


Figure 4.2: **The network structure of the garment auto-encoder**. The point cloud sampled from the original mesh is first fed to a global PointNet for coarse shape features. Its representative point set is then obtained by decoding the global features from an AtlasNet. From those points, I sample the local patches using K-nearest neighbor and pass them to a local PointNet for detailed shape features. The local decoder is then conditioned on the global latent code and the corresponding patch center to recover the patches that are stitched together to form the reconstructed point cloud.

not to assume fixed garment topology, I choose to use point clouds as the underlying representation. Other representations, such as graph-based [140] or displacement-based [10, 11], rely on either fixed graph structure, or fixed human surface, thus not applicable for generalization to different garments. The use of auto-encoder here is

necessary because the degrees of freedom (DoF) for point clouds are too high for estimation. An encoder-decoder structure can effectively reduce the DoF and retain only the essential information, such as the global shape and the local details. More importantly, it clusters similar shapes to similar latent codes, which is beneficial to the estimation module. As later shown in the Appendix, my model provides smooth transitioning between different topologies by using simple interpolation between latent codes, which is never achieved in previous works.

4.4.1 Two-Level Encoder-Decoder Structure

Previous point cloud auto-encoders such as AtlasNet [141] use Multi Layer Perceptron (MLP) to transform a 2D patch to a set of 3D points in the space. Their method performs well in point cloud datasets that include rigid objects, such as airplanes or chairs, since the deformations presented in those objects are simple and regular. However, it cannot be directly applied to learn garment point clouds, since garments have a much larger variance in point cloud distribution due to its dynamic nature. For example, a simple dress can create different wrinkle structures under different external forces. As a result, one global auto-encoder cannot account for all detailed structures, resulting in overly smoothed point clouds. Recently, [142] proposes a method to resolve patch overlapping and collapsing occurred in AtlasNet, but it still cannot account for arbitrary topologies and detailed wrinkles.

I propose a two-level auto-encoder for learning the latent space of the cloth. As shown in Fig. 4.2, I use a set of representative points $\mathbf{C}$ to express the global

shape of the garment, and sample around them to form local patches, which are encoded independently to account for local shapes. Specifically, given a point cloud $\mathbf{P}$, I first pass it through a global auto-encoder to form a representative point cloud:

$$\mathbf{C} = D_g(E_g(\mathbf{P}), \theta) \tag{4.1}$$

where E_g and D_g are the global encoder and decoder, and θ is the human body parameter. Next, I use K-nearest-neighbor to sample points around the representative ones:

$$\mathbf{P}_i = KNN(\mathbf{P}, \mathbf{c}_i) \tag{4.2}$$

where $\mathbf{c}_i$ is the i-th element in $\mathbf{C}$, and $\mathbf{P}_i$ is the i-th patch. This step forms local patches around the representative points. Finally, I pass each patch to the shared local auto-encoder, and do a union operation to obtain the reconstructed point cloud:

$$\mathbf{Q}_i = D_l(E_l(\mathbf{P}_i), \mathbf{z}_g, \mathbf{c}_i) \tag{4.3}$$

$$\mathbf{Q} = \bigcup_i \mathbf{Q}_i \tag{4.4}$$

where $\mathbf{Q}_i$ and $\mathbf{Q}$ are the reconstructed patches and point cloud, D_l and E_l are the local decoder and encoder, and $\mathbf{z}_g = E_g(\mathbf{P})$ is the global latent code.

4.4.2 Representative Point Set Extraction

Note that Eq. 4.3 and 4.4 imply that the representative points $\mathbf{C}$ have to be in the same order as the local latent codes $\mathbf{z}_l$. This is the key reason why traditional methods such as farthest point sampling [127] do not work: its ordering is very sensitive to the input, resulting in an unknown mapping between reconstructed patch centers $\mathbf{C}$ and the local patches $\mathbf{P}_i$ (thus the local latent code $\mathbf{z}_{l_i}$).

To resolve this issue, I encode the entire point cloud and compute the representative points using the decoder itself. Due to the continuous nature of the auto-encoder network, the continuity and consistency regarding similar point clouds are guaranteed, thus ensuring $\mathbf{c}_i$ to be exactly matched with $\mathbf{P}_i$.

4.4.3 Training Losses

During training, I use Chamfer Distance between two point clouds as the loss:

$$d(\mathbf{P}, \mathbf{Q}) = \frac{1}{|\mathbf{P}|} \sum_{\mathbf{p} \in \mathbf{P}} \min_{\mathbf{q} \in \mathbf{Q}} \|\mathbf{p} - \mathbf{q}\| + \frac{1}{|\mathbf{Q}|} \sum_{\mathbf{q} \in \mathbf{Q}} \min_{\mathbf{p} \in \mathbf{P}} \|\mathbf{q} - \mathbf{p}\| \tag{4.5}$$

I apply the Chamfer Distance loss between the representative point set and the point cloud to learn the global shape, and the one between the recovered and the original point cloud, both patch-wisely and globally, to capture the local details:

$$\mathcal{L}_{AE} = d(\mathbf{P}, \mathbf{C}) + \frac{1}{n} \sum_{i=1}^{n} d(\mathbf{P}_i, \mathbf{Q}_i) + d(\mathbf{P}, \mathbf{Q}) \tag{4.6}$$

4.4.4 Recovery from Point Clouds to Garment Meshes

The point cloud representation of a garment mesh does not explicitly store the connectivity information, so it is necessary to apply certain prior when recovering meshes from point clouds. One straightforward way is to connect each point with its neighbors, determined by a distance threshold. However, this method does not guarantee the resulting mesh to be a manifold. Instead, I use manifold surfaces to approximate the results.

The overall pipeline to recover point clouds to meshes is split into several steps. I first employ Screen Poisson algorithm [143] on the point cloud with estimated normals using neighboring points to recover the overall shape and topology of the garment. Next, I remove the reconstructed vertices that are too far away from the point cloud, since the first step tends to generate water-tight meshes. I focus on removing large clusters that forms holes for neck, arms, and legs. After upsampling on the resulting mesh, I deform the mesh by minimizing the distance $d_M(\mathbf{P}, \mathbf{F})$ between the point cloud $\mathbf{P}$ and the mesh $\mathbf{F}$ as defined below:

$$d_M(\mathbf{P}, \mathbf{F}) = \frac{1}{|\mathbf{P}|} \sum_{\mathbf{p} \in \mathbf{P}} \min_{\mathbf{f} \in \mathbf{F}} \hat{d}(\mathbf{p}, \mathbf{f}) + \frac{1}{|\mathbf{F}|} \sum_{\mathbf{f} \in \mathbf{F}} \min_{\mathbf{p} \in \mathbf{P}} \hat{d}(\mathbf{p}, \mathbf{f}) \tag{4.7}$$

where $\hat{d}$ is the point-to-face distance, $\mathbf{f} \in \mathbf{F}$ is a face in the garment mesh, $\mathbf{F}$ is the set of all faces of the mesh. I use other regularization terms similar to the mesh deformation demo in Pytorch3D [144].

4.5 Material Estimation

With the garment auto-encoder (Sec. 4.4) at hand, garment material estimation becomes tractable. I design my overall pipeline as shown in Fig. 4.3. Given the sequence of image frames, I first feed them one by one to a model for estimating the human body and cloth geometry. By predicting the latent vector instead of the exact positions of the point cloud, the single-frame estimation network avoids severe overfitting or producing irrational results, due to the reduction of the degree of freedom by the auto-encoder.

Next, I combine the image features as well as the estimated garment latent code as the temporal signals, which go through a canonical temporal network module (*i.e.* LSTM [145]) to predict the final material type. Since the latent space preserves similarity (*i.e.* positive correlation between distances of latent vectors and distances between the original point clouds), the motion of the estimated latent vector becomes a better indicator of garment motion than image features, which is beneficial to garment material learning. I do not include body features here because the garment material is directly related to the garment motion, which has already taken the human body as the condition (Sec. 4.5.1). I discuss more details of the network in the following sections.

4.5.1 Single Frame Closed-Loop Estimation

As shown in Fig. 4.4, I train a model to estimate the human body and cloth geometry given one single frame. Formally, in each frame, I am given the image

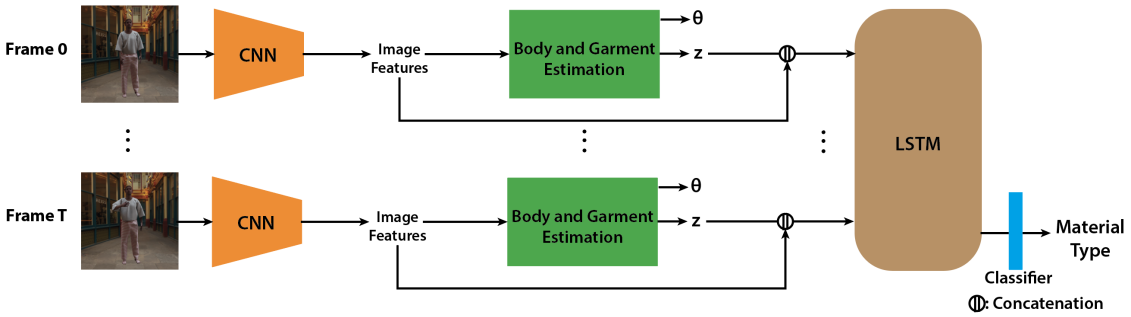
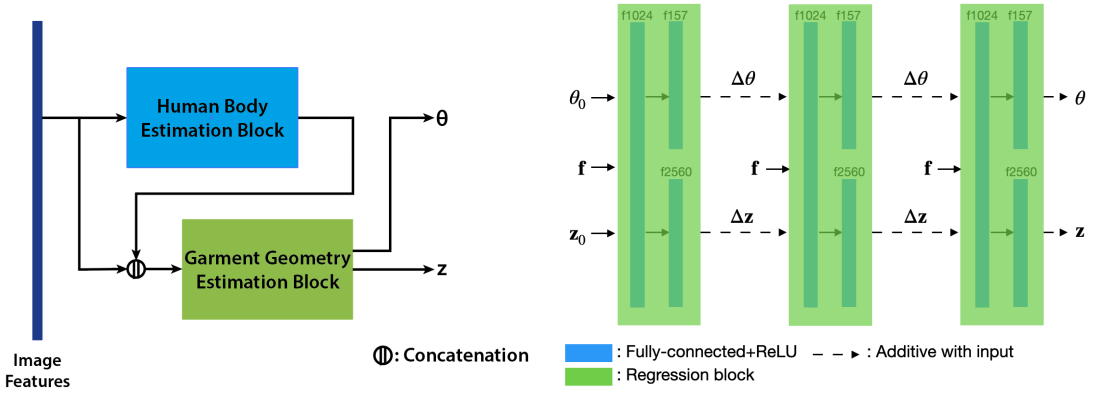


Figure 4.3: **My estimation pipeline**. Each video frame is first processed to obtain the image feature, the human body, and the garment shape. Then the image features are concatenated with the garment latent code as input to LSTM for material recovery.



(a) Closed loop structure for body & garment estimation

(b) Detailed structure in the garment estimation block

Figure 4.4: **The network structure for body and garment estimation in each frame**. (a) The garment shape estimation block takes the human body parameters as a prior, but also provides a feedback correction. (b) The garment estimation module consists of three identical, shared-weight blocks, each of which takes image features $\mathbf{f}$ and current predictions of the human body θ_0 and garment $\mathbf{z}_0$, and outputs the corrective values.

features $\mathbf{f}$. I first go through a state-of-the-art body estimation block, HB, to get a first-hand body estimation:

$$\hat{\theta} = HB(\mathbf{f}) \tag{4.8}$$

where $\hat{\theta} = [\theta, \beta]$ are the human body parameters including pose and shape. In the garment estimation block, I take as input $\mathbf{f}$ together with $\hat{\theta}$ and regress the garment latent code $\mathbf{z}$ and a final value of body parameters θ. Inside the garment estimation block, I use three shared-parameter small regression blocks, RB, to iteratively provide the correction, given the current estimation:

$$\theta_0 = \hat{\theta} \quad \mathbf{z}_0 = \mathbf{0} \tag{4.9}$$

$$\Delta\theta_i, \Delta\mathbf{z}_i = RB(\theta_{i-1}, \mathbf{z}_{i-1}) \tag{4.10}$$

$$\theta_i = \theta_{i-1} + \Delta\theta_i \quad \mathbf{z}_i = \mathbf{z}_{i-1} + \Delta\mathbf{z}_i \tag{4.11}$$

Overall, the garment estimation block forms a closed-loop structure, in which the human body parameters are required to predict the garment, and are later corrected back by the garment prediction as well.

The key insight of my module design is that the human body and garment shape are highly correlated at different scales and should be jointly learned using shared information. On the global scale, the detailed features of the garments restrict the variance of the human body and reduce ambiguity due to camera projection. On the local scale, the body pose and shape largely defines the valid distribution of the garment wrinkle positions. My proposed structure is also analogous to iterative optimization and feedback control in other areas, where two objectives serve as prior knowledge of each other and are improved iteratively. My work is the first to introduce this idea to the human and garment joint estimation task.

The loss function for the single-frame estimation is defined as:

$$\mathcal{L}_s = \mathcal{L}_{body} + \mathcal{L}_{AE} \tag{4.12}$$

$$\mathcal{L}_{body} = \mathcal{L}_{2D} + \mathcal{L}_{3D} + \mathcal{L}_{SMPL} \tag{4.13}$$

where $\mathcal{L}_{2D}$, $\mathcal{L}_{3D}$, and $\mathcal{L}_{SMPL}$ represent the 2D joint loss, the 3D joint loss, and the body parameter loss defined to supervise the human body estimation [136], and $\mathcal{L}_{AE}$ is the Chamfer distance defined in Eq. 4.6 to supervise the garment estimation.

4.5.2 Temporal Estimation for Garment Material

Garment material estimation is challenging since the visual difference of different materials is subtle and can easily be overwhelmed by disturbance, *e.g.* various directions or magnitudes of external forces. To tackle this problem, previous works often assume fixed environment settings and cloth shapes [21, 110]. While I follow a similar principle when training the material estimation module, I go one step further that I only assume common human motion for driving the garment instead of the whole external force field. While previous works [21, 110] can only handle videos of a piece of cloth hanging and dragged by the wind, my method possesses a wider applicability regarding the diversity of the garment shapes, sizes and human motions in the input video, which for the first time enables practical usages for garment material cloning.

As shown in Fig. 4.3, I collect and concatenate the image features and the estimated garment latent vector of each frame as the input signal, and feed the

sequence of the signals to LSTM to produce a summary feature. Finally, I pass the summary feature to a fully-connected layer for material type classification. I use the cross-entropy loss for supervision.

I train the model in an end-to-end fashion, but deliberately fix the single-frame estimation module. While my training does not benefit from end-to-end fine-tuning (which I technically can), it poses an even more challenging task for the temporal estimation module. No matter how expressive the network is, the training results will not be great, if the input signal provided by the previous module is noisy. My experiments demonstrated in Sec. 4.7 indicate that the multi-scale garment features are not merely useful for detecting the fabric materials; they are the dominant features during the estimation and can boost the test accuracy compared to methods that only feed the image features.

4.6 Data Preparation and Training

In order to train my model, a large number of examples that contain ground truth human body parameters, garment meshes, and the corresponding material parameters are needed. These are very challenging to capture in real world. To supplement a very limited number of such real-world videos, I create a large dataset of videos generated with controlled variables with the corresponding ground-truth values for validation. I vary different conditions to generate this dataset:

Human motion. I sample common human motion sequences and shape parameters in the CMU Mocap dataset [65], including walking, sitting, boxing and climbing

stairs.

Garment meshes. I use the garment dataset from [146].

Material space. I sample different materials uniformly in the discretized space mentioned in Sec. 4.3.

Human and garment textures. I use random human textures from SURREAL [147] and random garment textures from online images.

Lighting. I employ diverse outdoor and indoor environment maps downloaded from [148].

In total, I create a dataset of 250,000 images (10 motions * 100 garments * 250 frames) for single-frame human and garment geometry estimation, and 140,000 video sequences with each consisting of 25 images sampled at a frame rate of 5. Some examples of my training datasets are shown in Fig. 4.15.

4.6.1 Training Details

The split percentages for the train/validation/test sets are 85%/5%/10%. During training, I use Adam [149] to minimize the loss. The learning rate is 10^{-3} for the auto-encoder, and 10^{-4} for others. They are scaled down to 10^{-4} and 10^{-5} respectively after 10 epochs. All hyper-parameters are chosen empirically. As shown in Fig. 4.2, I use a representative point set of size 256, and generate 128 points for each patch. The sampling size for the ground-truth point set (N in Fig. 4.2) is 16,384.

I trained and tested my model on a machine with 8 CPUs (Intel Xeon, 3.60GHz) for data loading and 2 GPUs for computing (GeForce GTX 1080). I trained my auto-

encoder, single-frame module, and the temporal module for 20 epochs, respectively.

4.7 Experiments

I demonstrate the performance of my model as follows. (1) To illustrate my model's generality and effectiveness *quantitatively* (Sec. 4.7.1) and *qualitatively* (Sec. 4.7.2), I compare the results of my material estimation pipeline with the baselines and previous works. (2) I conduct a user study to investigate material perception using this work and quantify the similarity between the measurements of real-world fabrics from lab experiments and predicted garment materials from videos (Sec. 4.7.3). (3) I conduct additional comparison with other related learning-based methods, including ablation studies and latent code for garment interpolation/design, and application to virtual try-on.

4.7.1 Quantitative Analysis

Mine vs. Image-Only [21]. Due to the difference regarding the input distribution (dressed garment on a human body in my method vs. hanging cloth in theirs), I re-train their model on my datasets for a fair comparison. I study the contribution of image-only features vs. garment-only features, as well as CNN vs. LSTM (that exploits the temporal coherence). Finally, I compare the overall performance difference between mine and [21]. The test classification accuracy is reported in Table 4.1.

Findings: (1) While all three models have learned the relationship between

motion and materials and all three outperform random guess, *the garment feature signals are shown to be much more important than the image features.* This finding is not surprising, since the garment shape is directly affected by the material. (2) Combining the two features, as my model does, further improves the test accuracy. A possible reason is that an overall capturing of the garment shape (*e.g.* width and length of the entire piece), which is difficult to retrieve using garment latent codes, could be more easily extracted using image features. (3) By exploiting temporal coherence, unsurprisingly all three versions of the model achieve better accuracy than only using 1 image.

Method	Mean Accuracy	Temporal Gain	Garment Features Gain
Random guess	1.85%	-	-
Image only, CNN	5.11%	40.16%	-
Image only, LSTM [21]	45.27%		-
Garment only, CNN	11.85%	53.31%	6.74%
Garment only, LSTM	65.16%		19.89%
Image + Garment, CNN	12.62%	**57.52%**	7.51%
Image + Garment, LSTM **(mine)**	**70.14%**		**24.87%**

Table 4.1: **Comparison on material estimation**: my method achieves much higher accuracy (∼50%) in classification than [21].

Mine vs. Optimization-based [110]: An optimization-simulation framework to obtain the fabric material parameters using wrinkle density of the garment in a single image was proposed in [110]. In contrast, my method extracts both static image features and spatio-temporal *garment features* across frames. I generate the same set of test scenes as shown in Fig. 4.9. My model is tested on these sequences under varying lighting and visibility conditions; the average accuracy is reported in

Table 4.2. In this challenging case where the lighting condition and the textures are not seen in the training distribution, my method still achieves comparable accuracy with previous method [110], but it runs **more than 1,000x faster**. *Mine is the first end-to-end learning-based method to predict fabric materials directly from a video of garments worn on a human body.*

Method	Accuracy (%)						Speed
	Mid-day			Sunset			
	T-shirt	Pants	Skirt	T-shirt	Pants	Skirt	
[110]	80.2	80.2	**83.3**	81.6	79.9	**80.7**	4-6 hours
Mine	**86.5**	**91.6**	81.6	**82.4**	**91.6**	79.6	**8.7s**

Table 4.2: **Quantitative comparison with [110]**. My method achieves comparable or higher accuracy, but runs at least **three orders of magnitude** faster than the state-of-the-art [110].

4.7.2 Qualitative Results

I compare mine with the most relevant work of [110] for joint estimation of garment shapes and materials, as shown in Fig. 4.5. My method achieves similar reconstruction accuracy and visual quality as [110]. But, [110] uses semantic segmentation, thus suffering from tedious manual processing and long inference time. In contrast, my learning-based method is fully automatic and can compute the prediction in real time. Moreover, my method does not assume the sewing patterns as a prior.

I further compare with several learning methods [10, 118] in Sec. 4.7.6. Many often use additional information (*e.g.* mesh templates or known garment types) as priors, so direct visual comparison is not meaningful. Nonetheless, my model

successfully generalizes to unseen real-world images/videos with comparable visual results, as shown in Fig. 4.13. During these experiments, my method is directly applied without any fine-tuning or post-optimization. Although trained using synthetic datasets, my model correctly identifies people and the garments from real-world images, and achieves similar visual results in all examples, when compared with previous works. The network is also capable of the predicting correct sizes of garments relative to the body, due to multiscale auto-encoders.

Material Cloning for Virtual Try-On: I show three application scenarios of my method. In Fig. 4.6, given an RGB video of a person wearing garments of different fabric materials, my method can identify the underlying material and *clone* it onto other garment models using cloth simulation. My method is the first to achieve fast and accurate material extraction from videos of dressed garments on a body. I further show the ability of my method to reconstruct the entire human appearance from the input video using one single network. I first estimate the body and the garment geometry frame by frame, and use the temporal information to infer the material. The three parts are combined using cloth simulation to generate the final output. Fig. 4.7 and Fig. 4.8 show the reconstruction results (also see the supplement video). My reconstructed garment shapes and wrinkles match those in the input video frames.

(a) Input image (b) Results from [110] (c) My results

Figure 4.5: **Qualitative comparison with [110]**: Mine is easier to use and achieves visually comparable reconstruction much faster without priors on garment patterns and topology.

(a) Soft silk (b) Stiff polyester (c) Soft silk (d) Stiff polyester

Input videos **Estimated materials**

Figure 4.6: **Material transfer between videos**. My method can take videos of a person wearing any garments (a, b) and clone the underlying fabric materials onto other simulated garments (c, d).

(a) Real (b) Virtual (c) Real (d) Virtual

Figure 4.7: **Qualitative results:** my method faithfully recovers the T-shirt materials (a, c) in video so that the wrinkles around the simulated t-shirt sleeves (b, d) appear similar under different poses.

4.7.3 Lab Experiments and User Study

In this experiment, I test the prediction accuracy of my method using real-world materials. I used five real-world materials measured from lab experiments [93], which are sampled from sweater, t-shirt, tablecloth, jeans, and blanket, respectively. The materials are used to create videos using the same pipeline as I generate the training data. The videos are then input to my network model for material estimation, which is used to generate the resulting videos with other conditions being held the same (see Fig. 4.16 for sample frames).

I first quantify my material recovery using the sensitivity analysis described in [21]. For each of the materials, I measure its stiffness ratio according to the deformations under a fixed amount of external forces. The measured values are then compared with the one predicted from my method, reported in Table 4.3. My method achieves a relatively small error between 9.5% to 16.7%.

Material Name [93]	Stretching Ratio (GT/Prediction)	Bending Ratio (GT/Prediction)	Mean Relative Error
gray-interlock	1.01/1	1.6/2	10.5%
navy-sparkle-sweat	0.56/0.5	1.7/2	12.8%
white-dots-on-blk	15.8/20	3.5/4	16.7%
11oz-black-denim	3.6/3	3.0/3	8.3%
pink-ribbon-brown	2.93/3	12/10	9.5%

Table 4.3: **Lab experiment results.** My material estimation achieves relatively small errors compared to lab measurements on all real-world materials tested.

To further validate and quantify the material similarity, I conduct a user study to examine how close my estimation results are to the ground-truth data in human perception. In the study, I place two videos side by side for each material; then ask each participant to rate the level of material similarity: from 0 (totally different)

to 10 (identical). There are 25 subjects in my study group: 17 male and 8 female, with age ranging from 20 to 40. To further calibrate the subjective score range, I use a pair of videos generated from the *same material* but with *different mesh resolutions* and inform the participants that this example has a similarity score of 5 for calibration. My results show that the average similarity ratings for five tested materials vs. the ground-truth data are all larger than 5, ranging from 5.7 to 8.5, with an overall mean value of 7.1. These indicate that my method indeed can recover fabric materials with only minor perceptible differences to the real-world materials.

Besides the main results, I also conducted several other studies to investigate how people perceive the garment materials in different environmental conditions:

Garment color and texture. I changed the garment colors and textures; then asked participants the same questions. I found that by varying the garment colors, either brighter or darker, does not affect the similarity scores much – within a maximum difference of 0.4. On the other hand, changing the textures results in more perceptible effect – with similarity score differences of 0.5 to 0.9.

Lighting. I varied the lighting conditions in the rendered results to understand how shading on the garments affects material perception. The results show that the similarity scores are decreased by 1.1-1.3 when one of the videos has a different lighting angle than the other. These noticeable difference indicate the effects of lighting and shading on how humans perceive wrinkles and folds.

Stiffness range. I took the material called 'gray-interlock', consisting of 60% cotton and 40% polyester, and multiplied its material parameters by 1, 2, 5, and 10,

respectively. The participants were asked to distinguish which of the two sampled materials is stiffer. My findings indicate that there is little perceptible visual difference between lower stiffness values till when the garment stiffness is increased to a certain threshold. This finding also reconfirms my design rationale of the material space discretization using sensitivity analysis in Sec. 4.3.

Additional Results. I present more comparisons with other related learning-based methods for garment shape and pose recovery in Appendices. Please note that none of these methods was designed to *recover the fabric materials*, the main focus and motivation of this work for simulation-based virtual try-on, as shown in Fig. 4.8.

(a) Input video (b) Reconstruction result

Figure 4.8: **Qualitative comparison with a real-world video.**

4.7.4 Ablation Study

Method	CD	SD
Single-scale	0.31	8.05
Multi-scale	**0.12**	**1.03**

(a) Auto-encoder

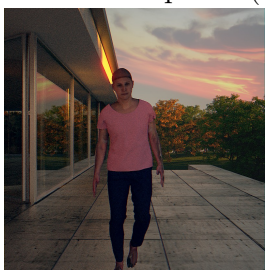
Method	MPJPE	CD
w/o feedback	62.93	0.89
w/ feedback	**55.20**	**0.88**

(b) Human and garment estimation

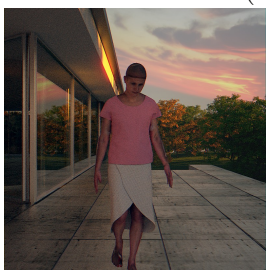
Table 4.4: **Ablation study for different parts of my proposed network.** My method is marked in Italic. CD stands for errors in Chamfer Distance; SD stands for Sinkhorn Divergence [150]; and MPJPE stands for Mean Per Joint Position Error – all in millimeters (mm). My method results in notably smaller errors than all baselines in the estimation and reconstruction tasks.

(a) T-shirt and pants (day)

(b) T-shirt and skirt (day)



(c) T-shirt and pants (night)



(d) T-shirt and skirt (night)

Figure 4.9: **Sample test images for the comparisons with [110].**

In the following ablation studies, I verify the effectiveness of my network model. I use the test errors to compare my model with the baselines, which include previous methods or their combinations. During the test, all other conditions are held the same, except for the network structure itself.

w=0.0 w=0.2 w=0.5 w=0.8 w=1.0

Figure 4.10: **Interpolation between different garments**. w is the interpolation weight. I extract the latent code of 2 different garments: a long-sleeve top & pants (leftmost) and a short-sleeve dress (rightmost). I use the decoder to produce new garments of linearly-interpolated latent codes, enabling smooth transitioning between topologically different garments not achieved before.

Garment auto-encoder. I use a two-level auto-encoder structure to address the highly dynamic geometry of garments, as discussed in Sec. 4.4. I demonstrate here

Figure 4.11: Material transfer examples. My method can accurately estimate the material parameter from the input video and replicate the same effect in other animations. Note that different estimated materials create significantly different visual appearance even for the same animation.

its effectiveness compared to the baseline, which simply consists of a PointNet [99] encoder and an AtlasNet [141] decoder. There are two theoretical advantages of my method against the baseline. First, the two-level structure partitions the entire point cloud by patches that only overlap by a small fraction. This approach makes the local features more focused on its local shape rather than on a more expanded surface. Next, the two-level network also offers more capability to express detailed features and prevents overly-smooth reconstructions.

In Table 4.4(a), I report my test errors compared to the baseline. In addition to Chamfer Distance as I used in the training, I also use Sinkhorn Divergence [150], which is a fast approximator for computing Earth Moving Distance between two distributions. While Chamfer Distance indicates the average distance between two

point clouds, Sinkhorn Divergence captures the density difference across space. The numbers in the table show that my method not only reproduces point clouds closer to the ground-truth but also has more evenly-distributed points due to my patch-wise partition during training. The results align with my theoretical analysis.

Garment geometry estimation. In Sec. 4.5.1, I proposed a closed-loop feedback structure to improve the estimation of both the human body and garment shape. I conducted an ablation study to show the difference introduced by this structure. My baseline is a two-branch estimation block that predicts the human body and the garment in parallel but sharing the image features as input. Although it does not have the feedback correction, it already benefits from multi-tasking which can extract useful common image features. I use Mean Per Joint Position Error (MPJPE) for human estimation metric, and Chamfer Distance for garment estimation.

As shown in Table 4.4(b), by introducing an extra feedback loop, my model results in a much smaller error on human body estimation. Interestingly, the garment estimation error is not greatly improved, possibly because the error comes more from local shapes (especially those occluded by the human) than the global one.

I further compare my method with a recent work [10] since their task is the most relevant to mine. I use the public dataset from Multi-Garment-Net [10], which consists of 95 scans of people. Nearly half of them are not suitable due to incorrect or incomplete garment labeling. I tested my model in the dataset without any fine-tuning and used their reported numbers for comparison. I use the Chamfer Distance

defined in Eqn. 4.5 as my test metric. As shown in Table 4.5, my model, without any fine-tuning or domain adaptation, achieves the smallest error – whether the ground-truth human body model is provided or not during reconstruction. Since my model is trained on a wider range of body poses and garment types than theirs while achieving better accuracy, it has shown to offer good generality to unseen inputs.

Methods	[68]	[10]		Mine	
	GT Pose	GT Pose	Full Pred.	GT Pose	Full Pred.
Pants (mm)	5.44	5.57	10.16	**1.58**	**3.08**
Short Pants (mm)	8.23	5.97	10.00	**4.92**	**5.69**
T-shirt (mm)	5.80	5.63	11.97	**1.67**	**3.08**
Shirt (mm)	5.71	6.33	9.05	**2.29**	**3.75**
Coat (mm)	5.85	5.66	9.09	**2.84**	**3.65**

Table 4.5: **Test errors on the Multi-Garment Net dataset [10]**. My method achieves the best estimation accuracy across all garment types, without any fine-tuning or reference body.

4.7.5 Latent Code Interpolation

To showcase the expressiveness of my learned latent space, I conducted an experiment generating new garments by interpolation. I encode two different garments to obtain their latent codes and linearly interpolate in between. I generate the new point clouds using the interpolated results accordingly. The visual results are shown in Fig. 4.10. My results show that the interpolations represent a smooth transition between the two original garments, creating new garment styles that are not seen before during training. Note that modeling long dresses or garments with different mesh structures are not achieved in previous works, which either use dis-

placement maps [10, 11] (thus not able to model dresses) or mesh-CNN for encoding local features [8] (thus not applicable to different mesh topologies). My method is the first to propose *a feature space that unifies garments of different topologies with different body poses and shapes*, which is the key component to accurate garment material estimation, as demonstrated in Table 4.1.

I provide more interpolation results in Fig. 4.12. As described in Sec. 4.7.5, my algorithm enables two garment meshes to be interpolated using their latent code to generate new garments. The first row shows an example where the dress is shorten at the bottom and extended at the sleeves gradually. I also show interpolation results in the point cloud form in Row 2-4. The second row is another example of transformation from dress to pants, and the last two examples demonstrate the ability to interpolate the garment with different body poses. In these two cases, the garment type is the same, but the body pose is changing. Although the poses in between are never seen in the training set, the interpolated garment point clouds follow the pose transition, showing very little interpenetrations with the body. Note that when the human legs are moving, the garment correctly deforms with the pose, close enough to be consistent with the body motion, while still being collision-free. See the supplementary video for the interpolation animation.

4.7.6 Additional Qualitative Results

I tested my model without any fine-tuning on the images provided in MGN [10] and DeepCap [118], as shown in Fig 4.13. Although my model is never exposed

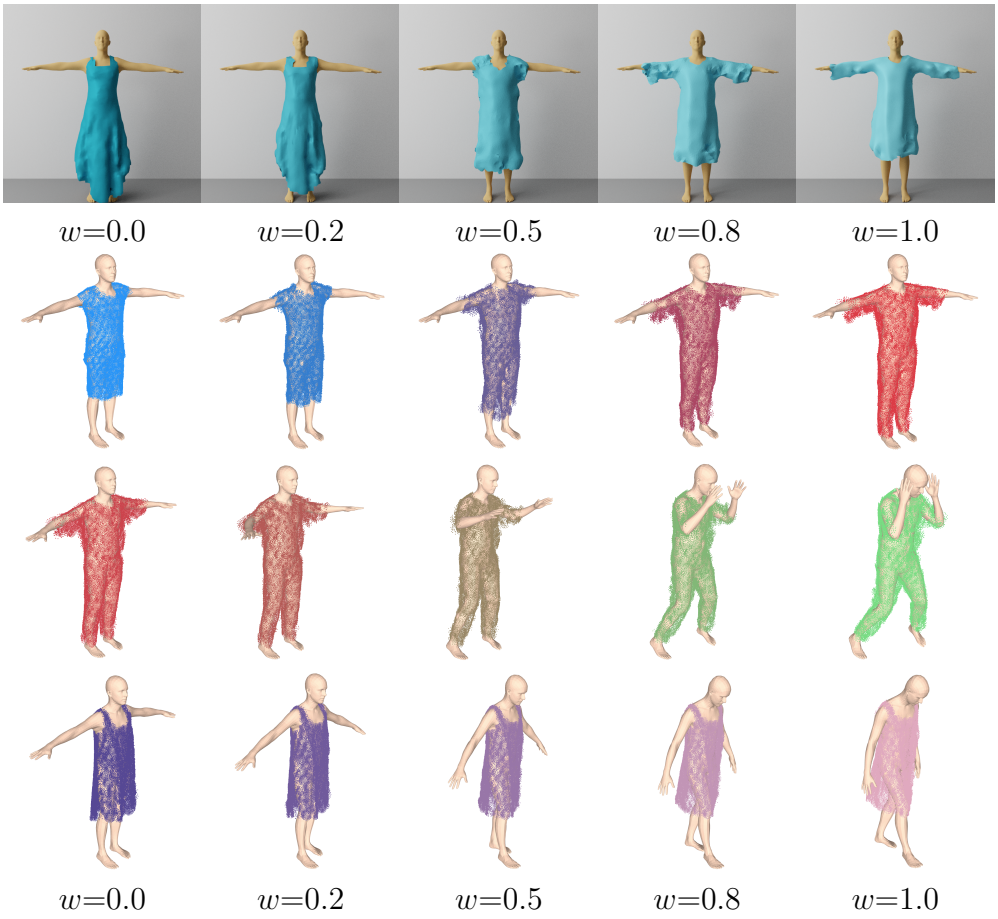


Figure 4.12: **Interpolation results**. My method can smoothly interpolate garments between different topologies and body poses.

to real-world images, it successfully predicts the human body pose and garment shape correctly. Note that my model is an end-to-end network that has no prior knowledge to any information about the garment shape, while MGN assumes that there is a one-to-one approximation mapping between body vertices and garment vertices. DeepCap has an initial ground-truth mesh that is to be optimized. My model can thereby easily generalize to unseen garments, which is not possible using these two works. In Table 4.6, I extensively compare my work with previous ones regarding different assumptions, functionalities, and abilities. I define 'one model per garment' in 'generality' as that the method needs to create extra templates or

registrations to the body, or need to retrain part of the network in order to predict a different garment type. Although DeepFashion3D [120] and ARCH [119] also have generality to different topologies to some extent, there are still limitations in their pipeline. The output from DeepFashion3D has to be continuous in one body part, meaning that they cannot support all topologies (*e.g.* dresses with holes). ARCH does support different garments on the body, but the output is a water-tight mesh together with the body, which is not always convenient for certain applications like virtual try-on. In contrast, my method naturally supports all kinds of topologies, and predicts the body and the garment in separate meshes.

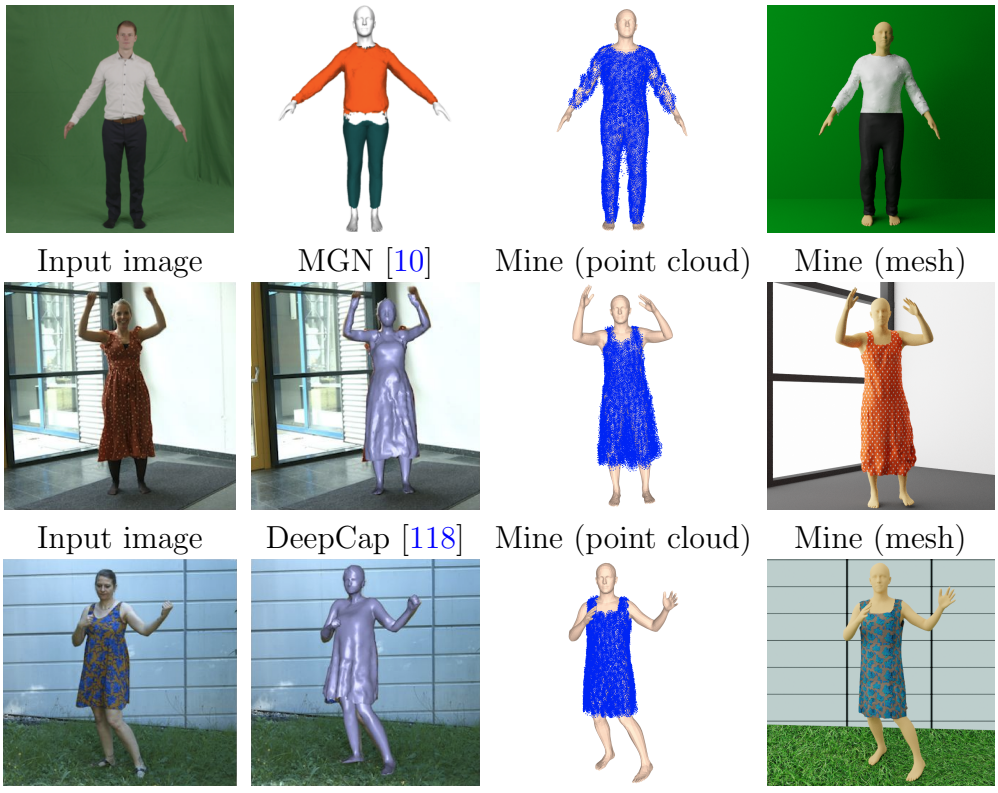


Figure 4.13: **Qualitative Results**. My model can achieve similar visual results with previous work without any knowledge of the target garment or any assumption of the topology (See Table 4.6).

Method	Input	Dependencies	Generality	Dresses support	Separate mesh	Material Estimation
MGN [10]	Semantic seg. + 2D joints	Garment correspondences	One model per garment	No	Yes	No
DeepCap [118]	Foreground seg.	Template mesh	One model per garment	Yes (w/ known template)	No	No
[110]	Semantic seg.	Template mesh	One model per garment	Yes (w/ known template)	Yes	Yes
DeepFashion3D [120]	RGB frame (garment only)	None	One model for all	Yes (limited topologies)	Yes	No
Tailornet [11]	Body parameters	Garment correspondences	One model per garment	Yes (limited topologies)	Yes	No
BCNet [121]	RGB frame	Garment correspondences	One model per garment	Yes (limited topologies)	Yes	No
SIZER [123]	Body scan	Garment labels	One model per garment	No	Yes	No
ARCH [119]	RGB frame (foreground only)	None	One model for all	Yes (water-tight)	No	No
Mine	RGB frame	None	One model for all	Yes	Yes	Yes

Table 4.6: **Comparison with previous works**. My method can handle the largest set of garments, using fewest possible information (*i.e.* RGB image), in one stand-alone, end-to-end network.

4.7.7 Application: Virtual Try-On

To further showcase the strength of my network, I apply it to a virtual try-on application. An online video clip showing a person wearing a dress is taken as input to my network (Fig. 4.14a). The body and the dress are estimated in each frame, and the fabric material is inferred using the garment motion and the image features. As shown in Fig. 4.14b, my method successfully infers the correct human body and the garment.

I then simulate the garment in a different body motion, which is the key functionality in virtual try-on systems. The simulated results (Fig. 4.14c) show that the garment motion provides similar visual impression with the input dress (mostly from the wrinkle motions of the dress). This example shows that my method can effectively extract the correct type of fabric material and transfer the given fabric material in a video to a simulation-based virtual try-on system. More animation results can be found in Fig. 4.11 and the supplementary video.

(a) Input video clip (b) Estimated result (c) Simulation result

Figure 4.14: **Virtual try-on example**. My network model can clone a person's appearance from the physical world (in a video) to the virtual world, enabling simulation under different motions with accurate estimations of the body shape, garment geometry, and fabric materials.

Figure 4.15: **Training data examples**. My dataset includes various garment topologies with rich body poses, textures, and background environments. Some examples are shown here.

4.8 Conclusion

In this chapter, I introduced an end-to-end learning model for garment material estimation using RGB videos. I do not assume other inputs (e.g. segmentation, 3D

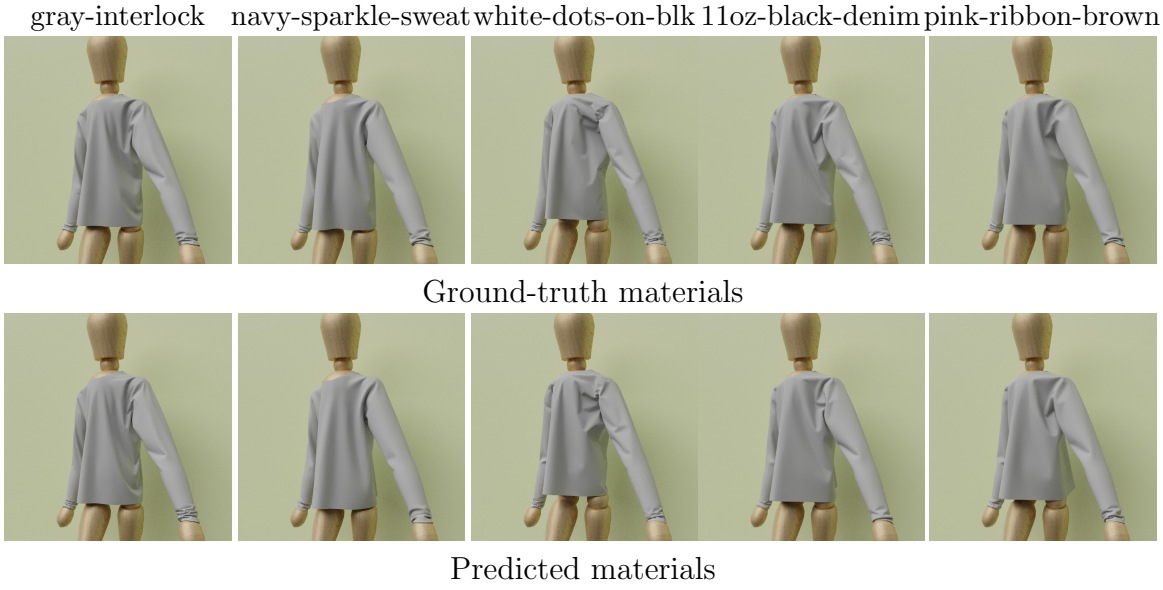


Figure 4.16: **User study examples**. My predictions received similarity scores of 5.7 to 8.5 in a 0-10 range.

scans, multi-views, etc.) or any prior knowledge on the garment shape/topology, design patterns/templates, or correspondences. I extract the multi-scale features to effectively represent the dynamic geometry structure of garments, which can be combined with image features to estimate fabric materials by learning their temporal patterns, while improving the human body reconstruction using a feedback loop. This approach is perhaps the first to introduce a unified parametric model for all garment types, and it can thereby support garments of different topologies without the need to retrain different models. Experiments show that my method achieves much higher accuracy up to 70.14% in estimating fabric materials than prior works, while offering capabilities in recovering garment types and topologies with generality and simplicity for an unification of multiple correlated tasks.

One limitation of this method is that the current representation does not support multi-layer or folded garments. These issues can be addressed by adding

structural prior to the garment model to encode multi-layer clothing and curvature representation to support multi-fold features. I further postulate that the proposed multiscale garment auto-encoder can also be integrated with neural rendering [151] to synthesize photorealistic images of simulated garments.

Chapter 5: Time-Domain Parallelization for Accelerating Cloth Simulation

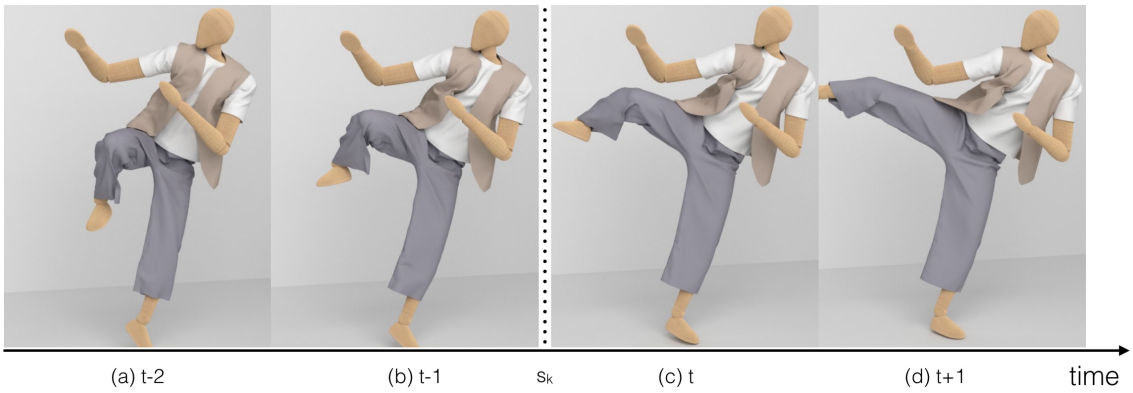


Figure 5.1: Simulated 'Karate' animation using my method. My method parallelizes the simulation workload in time domain using a two-level mesh representation. In the figure, the time domain partition point s_k is between frame t-1 and t, which will be simulated by two different processors. I use an iterative detail recovery algorithm to refine the state of the cloth from low-resolution mesh before the parallel high-resolution simulation begins. As a result, very little visual artifacts can be observed from (b) to (c). In the shown benchmark above, my parallelization method has achieved up to 99x speedup on 128-core systems – an unprecedented level of scalability in distributed CPU systems – compared to at most 47x on a 128-core system [152]. The performance gain is also better than the GPU parallelization [153] on similar benchmarks, while my approach offers the additional flexibility for coupling with adaptively remeshed cloth simulators.

5.1 Introduction

With the proper estimated body parameters, garment geometry, and fabric materials obtained from Chapter 4, it is now possible to synthesize garments on a

given sequence of body motion. One straight-forward way for garment synthesis is using cloth simulation. It offers realistic garment motion and collision-free results at the cost of high latency and throughput. Luckily, given the fact that the try-on system often resides on the cloud, it is possible to accelerate the cloth simulation by making use of manycore and cloud computing.

In this chapter, I propose a novel method for parallelizing cloth simulation. Unlike previous methods, my method divides the workload in *time* domain that minimizes the communication overhead, thereby achieving much better scalability and higher performance gain over previous methods.

The key challenge in time-domain parallelization is to obtain or approximate the simulation states before the time-consuming simulation begins. I use a two-level mesh representation to address this time-dependency issue. Observing that a coarse-level mesh can be simulated at a much higher speed, my method runs a lower-resolution simulation using coarser meshes to approximate the state at each time step. After an appropriate remeshing process, the higher-resolution simulations using finer meshes can be run in parallel. To further refine the simulation results, I propose a practical technique to smooth the state transition from the low-resolution to high-resolution simulations. To recover the lost states, I make use of the coarse-level mesh and run several 'static' simulation steps before the high-resolution simulation starts. Experiments in Sec. 5.6 show that this technique can reduce the visual artifacts between temporal partitions. In order to balance the workload of each processor, I further develop an adaptive partitioning algorithm, which takes into account the varying time consumption of each frame caused by

different contact configurations. I make use of the time measurements of previous frames in both mesh resolutions and determine the partition point based on the current estimation of the total running time.

To sum up, the key contributions of this work include:

- A time-domain parallelization algorithm supporting *adaptive meshes* with minimal communication overhead (Sec. 5.3);
- Load estimation and load balancing techniques that maximize the overall performance acceleration (Sec. 5.4);
- A practical state transitioning algorithm between low- and high-resolution simulations to recover details and ensure the visual quality of the simulated sequences (Sec. 5.5).

On a given set of benchmarks, my method achieves an unprecedented level of scalability in distributed CPU systems when compared to [152, 154]. Its performance gain is also higher than the GPU parallelization [153], while my approach offers the additional flexibility for coupling with adaptively remeshed cloth simulators. I also verify that given sufficient amount of processors, my method can achieve an average performance as fast as the low-resolution simulation, while obtaining simulation results similar to ones using high-resolution meshes. This method can be widely adopted in applications, where runtime performance is much more critical than accuracy, such as rapid design prototyping.

5.2 Related Work

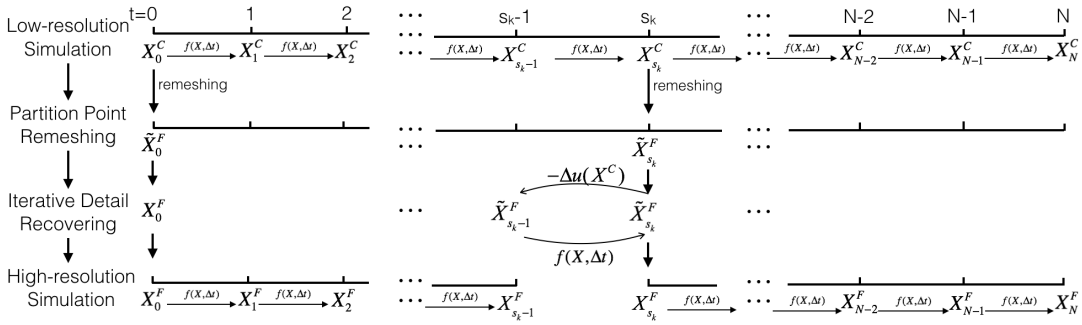


Figure 5.2: An overview of my method. I first simulate the cloth mesh in low resolution, obtaining the approximated states $\mathbf{X}_k^C$. After I select the starting point in time for each processor s_k (Sec. 5.4), I use the upsampling function to generate the initial states $\tilde{\mathbf{X}}_{s_k}^F$ and recover the detail information iteratively (Sec. 5.5). Lastly, I simulate the entire sequence in parallel, given the starting states $\mathbf{X}_{s_k}^F$.

In this section, I survey recent works on cloth simulation, parallelization techniques, and other related acceleration techniques for physics-based simulation.

5.2.1 Cloth Simulation

Simulation of cloth and deformable bodies has been extensively studied for a wide range of applications in different areas, from computer graphics, CAD/CAM, robotics and automation, to textile engineering. Due to their ability to take large time steps, implicit or semi-implicit methods [24, 155, 156, 157] have been widely adopted after the seminal work by Baraff and Witkin [27]. However, most of these works focus on the serial simulation improvement and their runtime performances can be slow. I use one of the state-of-the-art simulation algorithms, ARCSim [74], as the cloth simulator in my prototype implementation, but my parallelization technique does not rely on any specific simulation algorithm.

5.2.2 Time Parallel Time Integration Method

The scientific computing community have thoroughly studied parallelization techniques solving partial differential equations [158, 159, 160]. I refer readers to this survey by Gander et al. [161] for more details. Cloth simulation is similar to the general time-evolution equations. However, there is a gap for these works to be directly applicable. Cloth simulation has coupled other non-PDE factors, such as the collision response due to continuous contacts with the human body. The standard collision response within Physically-based Modeling literature is usually an "empirical" impulse applied mainly on the boundary cases, where the cloth is about to collide with the body or within a pre-defined 'threshold' neighborhood. Traditional solutions [158] use an arbitrary initial guess (e.g.$\mathbf{X}_t = \mathbf{X}_0$) for each of the time step and try to update the overall solution using a fixed point iteration. The discontinuity introduced by collision not only prevents the method from solving the fixed point problem in Newton's method (calculating derivatives of the conditional term determined by variables to be solved), but also prevents most of the collision response algorithm from obtaining stable and correct results (a severe inter-penetration of $\mathbf{X}_t = \mathbf{X}_0$ at time t that can hardly be handled). This special characteristic of cloth simulation makes it challenging to apply methods solving pure integrations (where the solution space is often regular) such as PFASST [158], due to collision-induced discontinuities.

5.2.3 Parallel Cloth Simulation

Several parallelization techniques for cloth simulation have been proposed. [162, 163] proposed GPU-based simulation methods for elastic bodies. [164, 165, 166, 167, 168] proposed different types of spatial parallelization but they all suffer from severe sub-linear scalability due to large communication overhead. [152] improved the work from [169] using Asynchronous Contact Mechanics and reduced the communication by proposing a locality-aware task assignment, which first scaled more than 16 cores. [153] implemented a GPU-based simulation pipeline. Their method has achieved an impressive speedup of 58 times, which is comparable to the performance of my method on a 64-core cluster.

The main difference between other parallelization methods and mine is that I decompose the simulation task in *time* domain. Partitioning in time domain significantly reduces the communication cost in distributed systems, thereby offering a considerable speedup. To the best of my knowledge, my method is the *first time-domain parallelization* algorithm for cloth simulation that can be coupled with *adaptive remeshing schemes*.

5.2.4 Hierarchical Structures and Multi-level Methods

Multi-level algorithms have offered significant performance improvement on various simulation problems. Tamstorf et al. [170] proposed a multi-grid method to speed up the cloth simulation. Bergou et al. [171] developed a tracking solver for rapid interaction in animation. They set up a two-level mesh representation and used

the desired coarse level animation to guide the fine level one by applying constrained dynamics. My method builds on top of their work to ensure the low-res consistency of the results. Recent works [172, 173, 174] generate high-resolution wrinkles from low-resolution cloth. My method is a physically-aware approach; it's more diverse and realistic compared to those work. Mine is more of an intermediate trade-off between time-consuming simulation and physically-unaware wrinkle synthesis. I use a hierarchical mesh representation to approximate the states of the cloth mesh at each time step, before transitioning to computationally expensive high-resolution simulations on fine meshes.

5.2.5 Mesh Upsampling

Mesh upsampling algorithms are widely explored from geometrical approaches [175, 176, 177] to data-driven methods [178, 179]. My method needs a specific mesh upsampling function to transfer the (approximated) state of the simulated cloth from low-resolution to high-resolution. While classic subdivision methods [177] cannot generate high-resolution details, data-driven ones [178, 179] depend largely on the specific configuration in the training data, and as a result, can generate interpenetrations when applying to arbitrary scenarios. For generality, I do not assume any specific upsampling function. Instead, I introduce an iterative detail-recovering approach described in Sec. 5.5 in order to account for the lost details in the low-resolution mesh. In my experiment, I use an adaptive remeshing method in [74] for its flexibility of use and a straightforward, linearly-interpolated subdivision for fast

error computation.

5.3 Overview

In this section I give an overview of my approach. I define the problem formally before I introduce the basic idea of the method.

Problem Statement: Given the initial state of a cloth mesh, $\mathbf{X}_0$ (inclusive of both position and velocity), generate a sequence of cloth states $\mathcal{V} = \{\mathbf{X}_1, \ldots, \mathbf{X}_N\}$ that characterize the cloth interaction with the given environment, using a time step Δt and a simulation function $\mathbf{X}_{k+1} = f(\mathbf{X}_k, \Delta t)$.

Fig. 5.2 shows the overall pipeline of my algorithm. The key idea of this method is to partition the time domain of the cloth simulation rather than the spatial domain of the simulated cloth. In order to obtain the (approximated) mesh state without full simulations, I propose a two-level hierarchy representation. I simulate the cloth mesh $\mathbf{X}^C$ at a coarser level with much lower computation and determine the partition point S (in time) according to the algorithm described in Sec. 5.4 before I simulate the entire high-resolution sequence $\mathbf{X}^F$ at the finer level in parallel.

The fine-level mesh at the starting point of each temporal partition is obtained by the corresponding coarse-level mesh using an upsampling/remeshing function $u(\mathbf{X}^C)$. However, the finer mesh may be quite different from the coarse one after remeshing because high frequency information $\mathbf{X}^D$ is not stored in the coarse-level mesh. Therefore, I design a practical state-transitioning technique to recover the

lost details to the extent possible, before the high-resolution simulation begins. This state-transitioning method will be discussed in Sec. 5.5. I list the notations used in this chapter in Table 5.1.

NOTATION	DEFINITION
$\mathbf{X}_k$	state of the cloth at step k
$\mathcal{V}$	output sequence of states
N	simulation sequence length
Δt	specified time step
$f(\mathbf{X}_k, \Delta t)$	one-step simulation
$f^i(\mathbf{X}_k, \Delta t)$	i-step simulation
$\mathbf{X}^C$	coarse level state
$\mathbf{X}^F$	exact fine level state
$\mathbf{X}^D$	state difference between the two level states
$\tilde{\mathbf{X}}^F$	approximated fine level state
$u(\mathbf{X}^C)$	upsampling function
p	number of processors
S	ordered set of starting points for parallelization
s_j	starting point of the jth processor
K	coarse-to-fine ratio

Table 5.1: **Notations and definition of my method.**

5.3.1 Two-Level Mesh Hierarchy Representation

Ideally I want to divide the whole simulation process into several temporal partitions so that I can simulate each partition in parallel and independently. However, since the mesh state at step k, $\mathbf{X}_k$, is determined by the state at previous step $\mathbf{X}_{k-1}$, I do not know the exact intermediate states until I finish the simulation from step 0 to step k. Here I use the hierarchical mesh representation to address this time-dependency problem. I maintain two sets of simulated meshes, $\mathbf{X}^C$ and $\mathbf{X}^F$, which represent the low- and high-res(olution) simulation states using the coarse- and fine-level meshes, respectively. I can recover the high-res state from the low-res

one by a user-defined upsampling function: $\tilde{\mathbf{X}}^F = u(\mathbf{X}^C)$.

Note that the obtained high-res state from the fine mesh, $\tilde{\mathbf{X}}^F$, is only an approximation of the exact state $\mathbf{X}^F$. But, for simplicity, I assume that $\mathbf{X}^F = \tilde{\mathbf{X}}^F$ in this section. Further state refinement is discussed in Sec. 5.5.

Due to the fact that the simulation using a coarse mesh is significantly faster than the one using a fine mesh, I can obtain low-res states $\{\mathbf{X}_1^C, \ldots, \mathbf{X}_N^C\}$ in a relatively small amount of time. I further choose p starting points $S = \{s_0 = 0, s_1, \ldots, s_{p-1}\}$ in time for p processors, according to my partitioning algorithm to be discussed in Sec. 5.4.1, and run the high-res simulation using the fine mesh in parallel:

$$\mathbf{X}_k^F = \begin{cases} \tilde{\mathbf{X}}_k^F & k \in S \\ f^{k-s_j}(\mathbf{X}_{s_j}^F, \Delta t) & s_j < k < s_{j+1} \end{cases} \tag{5.1}$$

where

$$f^i(\mathbf{X}_k, \Delta t) = \begin{cases} f(f^{i-1}(\mathbf{X}_k, \Delta t), \Delta t) & i > 1 \\ f(\mathbf{X}_k, \Delta t) & i = 1 \end{cases} \tag{5.2}$$

for running i steps of simulation.

5.4 Time Domain Parallelization

In this section I will describe my parallelization technique. I solve the partitioning problem from the simplest case to the most complex one, in order to balance the workload of each processor.

5.4.1 Static Temporal Partitioning

A straightforward approach for the partition problem is to divide the time domain into p temporal segments of the same length:

$$s_j = \lfloor \frac{N}{p} j \rfloor \tag{5.3}$$

Assuming that every simulation step using the fine mesh takes the same amount of time, the overhead of this partition schedule is the time spent in simulation using the coarse mesh. To further simplify the case, I take another assumption that the simulation speed at the low-res level is K times as fast as high-res level. I can estimate the speedup as:

$$\eta_1 = \frac{KN}{K\frac{N}{p} + (p-1)\frac{N}{p}} = \frac{Kp}{K+p-1} \tag{5.4}$$

Note that in the low-res simulation using a coarse mesh there is no need to continue the simulation after I reach s_{p-1}. Therefore, the time spent on low-res simulation is $(p-1)\frac{N}{p}$.

One improvement of the straightforward approach is that I can start the high-res simulation in parallel, as long as the corresponding starting point is ready. Intuitively, I want all processors of the system to finish their jobs at the same time to achieve a good workload balance and the best speedup possible. This objective can be attained by adjusting the starting points so that the processor which starts

earlier takes a longer part to simulate. Taking the same assumption, I arrive at a load-balancing equation:

$$s_j + K(s_{j+1} - s_j) = s_{j+1} + K(s_{j+2} - s_{j+1})$$

Recall that K is the ratio between the high- to low-res simulation time, s_j, s_{j+1}, and s_{j+2} are the starting point for simulation on the processors $j, j+1$, and $j+2$, respectively. This equation yields:

$$s_j = \lfloor \frac{1-q^j}{1-q^p} N \rfloor \tag{5.5}$$

where $q = 1 - \frac{1}{K}$. The speedup can then be expressed as:

$$\eta_2 = \frac{KN}{K(s_1 - s_0)} = K - K(1 - 1/K)^p \approx p - \binom{p}{2}\frac{1}{K} \tag{5.6}$$

This is a tighter bound than Eqn. 5.4, as p approaches to K. The key reason behind the sub-linear speedup is that the overhead ratio to the original computation is $1/K$. In practice, the ratio between high- to low-res simulation time can be controlled by the user and can usually reach 100∼200 using the method described in Sec. 5.4.3, which is sufficient for running on a large distributed system.

5.4.2 Adaptive Partitioning

In the discussion above, I consider K as a known constant throughout the entire simulation process. However, it is highly unlikely that this would be the case. First of all, remeshing in the simulation run leads to a varying number of vertices and thus a dynamically changing size of the linear system. Secondly, the computational cost can vary considerably, even with the same mesh size, due to collision queries. Recent studies [153] show that collision detection and response can take up to 80% of the total running time. Moreover, the difference of per-step runtime is also dominated by the collision response and the size of the adaptive mesh, which are largely related to the object granularity. It has much more impact in the high-resolution than the low-resolution, which K accounts for as well. Therefore, the ratio of high- to low-res simulation time varies and the exact number is usually unknown.

A fixed partitioning scheme can become unstable and sensitive to these variations, resulting in load imbalance. One common solution is to cut down the jobs into more smaller tasks so that the imbalance can be reduced by dynamic job scheduling scheme. This method surely works, but it will have large extra overhead due to job scheduling and required preprocessing time (Sec. 5.5), and extra hand-tuned granularity parameter to optimize the performance. Since I want to avoid any unnecessary computational overhead, I here propose an adaptive partitioning algorithm.

Suppose that I have simulated up to step n using the coarse mesh, when the first high-res parallel simulation with the same starting time has completed m steps,

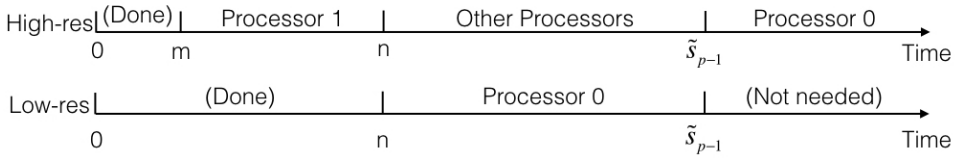


Figure 5.3: Adaptive partitioning Algorithm. I estimate the ratio of high-to-low-res simulation time, $\tilde{K}$, according to the runtime data I observe so far ([0,m] in High-res on Processor 1 and [0,n] in Low-res on Processor 0). The objective is to predict the future running time (marked by 'Processor 0' and 'Processor 1' respectively) to be as close as possible to the actual time.

where $m < n$. Let $T_C(m)$ and $T_F(m)$ denote the running time of the previous m steps using the coarse and fine meshes, respectively. Then, the ratio of the high-to-low-res simulation time, $\tilde{K}$, can be approximated as:

$$\tilde{K} = \frac{T_F(m)}{T_C(m)} = \frac{T_C(n)}{T_C(m)} \tag{5.7}$$

Since these numbers may vary, it is not appropriate to determine the global partition points using current approximations. Instead, I use them to determine if I should perform a cut on step n, i.e. whether n should be s_1 or not. Fig. 5.3 gives a visualization of the process. The objective of the partitioning algorithm is that the total running time on the processor 0, which performs the low-res simulation and the last part of the high-res simulation, is equal to the running time of the current parallel simulation that performs the high-res simulation using a fine mesh from step 0 to step n. This relation can be formulated as:

$$T_C(\tilde{s}_{p-1}) + (T_F(N) - T_F(\tilde{s}_{p-1})) = T_F(n) \tag{5.8}$$

where $\tilde{s}_{p-1}$ is the estimated starting point of the last partitioned segment. I use

the method described in Sec. 5.4.1 to obtain this parameter. I further approximate Eqn. 5.8 to:

$$n = \frac{N}{\tilde{K}} + \frac{\tilde{K}-1}{\tilde{K}}(N - \tilde{s}_{p-1}) \tag{5.9}$$

by assuming stable parameters in the remaining simulation:

$$T_F(j) = \tilde{K}T_C(j) = \tilde{K}T_C(1)j \quad \text{for any j} \tag{5.10}$$

Since n is increasing while $\tilde{K}$ and $\tilde{s}_{p-1}$ can be considered stable compared to n, Eqn. 5.9 can be defined at some point in $1 \leq n \leq N - p$. The remaining cut can be completed recursively. Algorithm 2 shows the pseudocode of this method. $\tilde{K}$ and $\tilde{s}_{p-1}$ here are approximated values used only for this cut. They can vary during the simulation, which will guide my partition algorithm to have adaptive cuts, instead of fixed ones in Sec. 5.4.1.

Algorithm 2: - Adaptive Partitioning

Require: N, p, X_0^C
1: $n \leftarrow 0$
2: start fine level simulation from step 0 on Processor 1
3: **while** true **do**
4: $n \leftarrow n + 1$
5: obtain X_n^C from X_{n-1}^C
6: $m \leftarrow$ steps finished by Processor 1
7: calculate $\tilde{K}$, $\tilde{s}_{p-1}$ from Eqn. 5.5 and 5.7
8: **if** condition of Eqn. 5.9 is met **then** break
9: **end if**
10: **end while**
11: $t_1 \leftarrow n$
12: Control Processor 1 to stop at Step n
13: Recursively partition remaining N-n steps with p-1 processors

In practice, the overall performance using adaptive partitioning is similar to that using static partitioning when the user can manually select the best K value for the simulation scenario. This algorithm generally offers the advantage of dynamically estimating the ratio of the high-to-low-res simulation time, so the user does not need to hand-tune this parameter for the best possible speedup.

5.4.3 Analysis on Performance Scalability

As discussed in the previous sections, the scalability of this time-domain partitioning method for parallel cloth simulation depends largely on the general runtime ratio between the high- to low-res simulation time, K. Since I perform a low-res simulation using a coarse mesh and a parallel one using a fine mesh, the low-res running time is a computational overhead for all processors and thus the speedup before any improvement is $\frac{K}{1+K/p} = \frac{Kp}{K+p}$. The ideal case of perfect workload balance, η_2, is discussed in Sec. 5.4.1, hence the actual performance of Algorithm 2 in a specific scenario, η_3, has the following theoretical bound:

$$\frac{Kp}{K+p} < \eta_1 \leq \eta_3 \leq \eta_2 < K \tag{5.11}$$

Therefore, the higher the K value is, the higher the overall performance gain of my method would be. One common way to increase K is to control the number of total mesh triangles by limiting the smallest possible size of each triangle in the low-resolution level. The other way is to enlarge the time step of the low-res simulation, since it is the common overhead of all processors and should aim for faster speed

rather than smaller discretization errors. A properly chosen large time step can improve the overall performance with minimal impact on the simulation results. With the coarsening techniques in space and time domains, K can be sufficiently large to obtain good scalability in large distributed systems.

5.5 Smooth State Transitioning

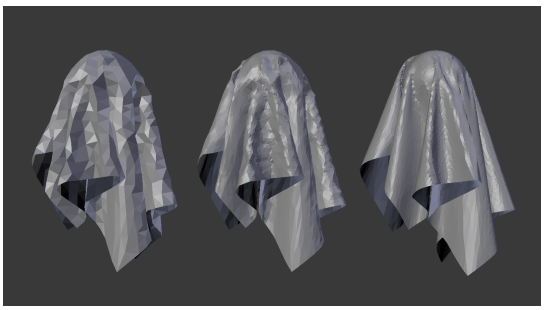


Figure 5.4: An example of the coarse mesh $\mathbf{X}^C$, intermediate mesh, $\tilde{\mathbf{X}}^F$, and the fine mesh, $\mathbf{X}^F$, after iterative corrections.

As mentioned in Sec. 5.4, the high-res simulation state approximation $\tilde{\mathbf{X}}^F = u(\mathbf{X}^C)$ is not the same as the exact state $\mathbf{X}^F$ using the fine mesh, the reason of which is that the high frequency information needed to reconstruct the states of the fine mesh is missing in the estimated states of the simulation using the coarse mesh. Therefore, if I take $\tilde{\mathbf{X}}^F$ directly as the starting state of the parallelized simulation, error $\mathbf{e} = E(\tilde{\mathbf{X}}^F, \mathbf{X}^F)$ will occur, since the high-frequency information is lost. Although $\mathbf{e}$ will vanish as the detail of the mesh is recovered by the simulation, another error will appear at the beginning of the subsequent partition after the end of the current one. (Here I focus on the actual visual effect instead of the L2 distance of each vertex. The error of my specific goal can be defined as the smoothness

of the cloth.) Thus, this error will appear as a 'popping visual artifact' in the final concatenated sequence of the cloth simulation. Fig. 5.4 shows an example of the inaccurate starting mesh (middle) obtained from the corresponding coarse level mesh (left), which causes a popping visual artifact because the error compared to the actual state (right) is large enough to be visible.

One straight-forward method is to apply global smoothing optimization as a post-processing step. However, this space-time optimization is too time consuming to be used in speed demanding applications. As mentioned before, Bergou et al. [171] used constrained dynamics for fine level simulation to match with the coarse level motion. I employ this method to prevent the high-res simulation from diverging too far from the low-res one. However, the high-frequency detail information would be still missing at the transition point. Inspired from the observation that the visual error will be eliminated during the simulation, I propose an iterative refinement technique that can recover as much as possible the high-frequency detail of the cloth from the low-res simulation using the coarse mesh.

5.5.1 Iterative Detail Recovery

Consider the mesh state at the consecutive step points $\mathbf{X}^C_{k-1}$ and $\mathbf{X}^C_k$. The fine-level mesh can be regarded as the sum of the low-frequency coarse mesh and the high-frequency detail:

$$\mathbf{X}^F = u(\mathbf{X}^C) + \mathbf{X}^D \tag{5.12}$$

Assuming that the time step is sufficiently small and the detail does not change

much between two simulation steps, I have:

$$\mathbf{X}^F_{k-1} - u(\mathbf{X}^C_{k-1}) \approx \mathbf{X}^F_k - u(\mathbf{X}^C_k) = \mathbf{X}^D_k \tag{5.13}$$

The idea here is to approximate $\mathbf{X}^F_{k-1}$ using $\mathbf{X}^C_{k-1}$, $\mathbf{X}^C_k$ and $\tilde{\mathbf{X}}^F_k$. From Eqn. 5.13 I have:

$$\tilde{\mathbf{X}}^F_k = f(\tilde{\mathbf{X}}^F_{k-1}, \Delta t) \tag{5.14}$$

$$\approx f(\tilde{\mathbf{X}}^F_k - u(\mathbf{X}^C_k) + u(\mathbf{X}^C_{k-1}), \Delta t) \tag{5.15}$$

Note that Eqn. 5.15 can be considered as an updated version of Eqn. 5.14. By subtracting the upsampled change of the state as a backward step and the simulation itself as a forward one, I can compute $\tilde{\mathbf{X}}^F_k$ iteratively. Algorithm 3 below shows the iterative detail recovery process. I run this algorithm at each of the transition point as a pre-processing step before the high-res simulation begins.

Algorithm 3: - Iterative Detail Recovery

Require: $\mathbf{X}^C_{k-1}$, $\mathbf{X}^C_k$ $(k \in S)$

1: $\tilde{\mathbf{X}}^F_k \leftarrow u(\mathbf{X}^C_k)$
2: **while** not reaching maximum iteration **do**
3: $\quad \tilde{\mathbf{X}}^F_{k-1} \leftarrow \tilde{\mathbf{X}}^F_k - u(\mathbf{X}^C_k) + u(\mathbf{X}^C_{k-1})$
4: $\quad \tilde{\mathbf{X}}^F_k \leftarrow f(\tilde{\mathbf{X}}^F_{k-1}, \Delta t)$ with constraints introduced by TRACKS [171]
5: **end while**
6: $\mathbf{X}^F_k \leftarrow \tilde{\mathbf{X}}^F_k$

5.5.2 Convergence and Continuity

Taking the advantage of the constraint-based tracking solver introduced by Bergou et al. [171], this iterative algorithm can be proved to have convergence guarantee. I show the proof in the following section. It is not guaranteed that the convergence point is exactly the same as the high-res simulation result. However, due to the enforcement of the tracking constraint, the difference compared to the result at the previous step will be $O(\Delta t)$, which means that there will be very little discontinuity and in most practical cases they are invisible.

5.5.3 Proof of Convergence of Algorithm 3

Theorem 1. *Algorithm 3 can reach the convergence point when applying the coarse-level tracking constraints to the system, as long as* $\frac{\partial \mathbf{F}}{\partial \mathbf{X}} = 0$ *for external forces.*

Proof. I assume the whole system is running under the Forward Euler method:

$$\begin{pmatrix} \Delta \mathbf{x} \\ \Delta \mathbf{v} \end{pmatrix} = \Delta t \begin{pmatrix} \Delta \mathbf{v} \\ M^{-1}\mathbf{F}(\mathbf{X}) \end{pmatrix} \tag{5.16}$$

where $\mathbf{F}$ is the force function, and $\mathbf{X} = \begin{pmatrix} \mathbf{x} & \mathbf{v} \end{pmatrix}^T$ is the state of the cloth. Given the assumption that $\frac{\partial \mathbf{F}}{\partial \mathbf{X}} = 0$ for external forces, they have the same contributions for each iteration and are all canceled out by the subtraction $(\Delta u(\mathbf{X}_k^C))$ in Algorithm 3. So I only consider internal forces.

Since I only focus on one high-res simulation step here, I leave off the resolution

superscript and replace the step number subscript by the iteration time. I denote the upsampled coarse-level difference by $\Delta\mathbf{X}_0 = \begin{pmatrix} \Delta\mathbf{x}_0 & \Delta\mathbf{v}_0 \end{pmatrix}^T$. Using the new notation, I have:

$$\begin{pmatrix} \mathbf{x}_i \\ \mathbf{v}_i \end{pmatrix} = \begin{pmatrix} \mathbf{x}_{i-1} - \Delta\mathbf{x}_0 \\ \mathbf{v}_{i-1} - \Delta\mathbf{v}_0 \end{pmatrix} + \Delta t \begin{pmatrix} \mathbf{v}_{i-1} - \Delta\mathbf{v}_0 \\ M^{-1}\mathbf{F} \end{pmatrix} \tag{5.17}$$

I now regard the evolution from $\begin{pmatrix} \mathbf{x}_{i-1} & \mathbf{v}_{i-1} \end{pmatrix}^T$ to $\begin{pmatrix} \mathbf{x}_i & \mathbf{v}_i \end{pmatrix}^T$ as one full simulation step (instead of a backward-forward iteration), and only focus on the velocity equation (since the position can be derived from it):

$$\mathbf{v}_i = \mathbf{v}_{i-1} + \Delta t(M^{-1}\mathbf{F} - \Delta\mathbf{a}_0) \tag{5.18}$$

where $\Delta\mathbf{a}_0 = \Delta\mathbf{v}_0/\Delta t$ is the corresponding acceleration value. Given that the internal forces are negative gradients of the potential energy, I have:

$$\frac{d^2\mathbf{x}}{dt^2} = M^{-1}\mathbf{F} - M^{-1}M\Delta\mathbf{a}_0 \tag{5.19}$$

$$= -M^{-1}\frac{\partial E}{\partial \mathbf{x}} - M^{-1}\frac{\partial M\Delta\mathbf{a}_0 \cdot \mathbf{x}}{\partial \mathbf{x}} \tag{5.20}$$

$$= -M^{-1}\frac{\partial E}{\partial \mathbf{x}} - M^{-1}\frac{\partial E_0}{\partial \mathbf{x}} \tag{5.21}$$

$$= -M^{-1}\frac{\partial \tilde{E}}{\partial \mathbf{x}} \tag{5.22}$$

where I make up a form of potential energy (E_0) with constant gradients to unite the two components.

By computing the dot product with the velocity (of the previous iteration), I have:

$$\frac{d\mathbf{x}}{dt} \cdot M\frac{d^2\mathbf{x}}{dt^2} = -\frac{d\mathbf{x}}{dt}\Big|_{(i-1)\Delta t} \cdot \frac{\partial \tilde{E}}{\partial \mathbf{x}}\Big|_{\mathbf{x}_{i-1}-\Delta\mathbf{x}_0} \tag{5.23}$$

$$= -\frac{d\mathbf{x}}{dt} \cdot (\frac{\partial \tilde{E}}{\partial \mathbf{x}}\Big|_{\mathbf{x}_{i-1}} - \frac{\partial^2 \tilde{E}}{\partial \mathbf{x}^2}\Delta\mathbf{x}_0) \tag{5.24}$$

$$= -(\frac{\partial \tilde{E}}{\partial t} - \frac{\partial^2 \tilde{E}}{\partial \mathbf{x}\partial t}\Delta\mathbf{x}_0) \tag{5.25}$$

$$= -\frac{\partial}{\partial t}(\tilde{E} - \frac{\partial \tilde{E}}{\partial \mathbf{x}}\Delta\mathbf{x}_0) \tag{5.26}$$

$$= -\frac{\partial \tilde{E}}{\partial t}\Big|_{\mathbf{x}_{i-1}-\Delta\mathbf{x}_0} \tag{5.27}$$

or in a discrete form:

$$\mathbf{v}_{i-1} \cdot M\mathbf{a}_i = -\frac{\partial \tilde{E}}{\partial t}\Big|_{\mathbf{x}_{i-1}-\Delta\mathbf{x}_0} \tag{5.28}$$

This equation means that the whole system tends to decrease the sum of the potential energy: when $\tilde{E}$ is decreasing, the acceleration $\mathbf{a}_i$ will have roughly the same direction with the velocity; otherwise it will have the opposite one, makes the velocity direction turn around eventually. The coarse-level tracking constraint here serves as a damping component, which prevents the system from oscillation due to conservation of energy. It also prevents $\tilde{E}$ from infinitely decreasing since the coarse shape of the mesh is strictly preserved [171]. Therefore, after sufficient number of iterations the whole system will reach a balance where $\frac{\partial \tilde{E}}{\partial t} = 0$, and a stable result gives $\mathbf{v}_i = \mathbf{a}_i = 0$.

□

Note that although I have constraints on external forces, in most of the cases,

they can be easily satisfied, such as gravitational forces and user-control impulse forces. Here I consider collision response as part of the constraint system, so it does not have impacts on the practical correctness. I use Forward Euler only for the simplicity of the expression in the proof. Actually I can derive the same form of Eqn. 5.18 using any other integrator (e.g. Backward Euler), during which the extra terms related to $\Delta\mathbf{v}_0$ (introduced by Backward Euler [27]) can be canceled out, eventually leaving $\Delta\mathbf{a}_0$. The main idea of the proof is that the system is conservative, regardless of the actual integrator, before adding extra damping constraints that ensures the final convergence. Upon convergence, the change in the high-res states (i.e. velocities and accelerations) will be the same as the change in the interpolated low-res states. This step, together with the position constraints by TRACKS, ensures the position and velocity difference between the high-res results at the boundary to be $O(\Delta t)$, smoothing out the visual popping artifact.

5.5.4 Iteration Number Estimation

The number of iterations needed for convergence, according to the proof, is largely related to the strength of the coarse-level constraint (in other words, the coarse-to-fine ratio K), since it provides the damping force to the system. Additionally, given a fixed upsampling scale (K), the iteration number is also related to a) the stiffness and density of the cloth, and b) the time step Δt. I estimate my iteration number in a simplified 2-D spring-mass system. Suppose at $t = 0$ a string with length l is hanging horizontally, with both endpoints fixed. It is currently

discretized as one single piece of 1-D string so the middle part of itself will not fall down. However, in the continuous real-world space, it is not in the equilibrium state and it has a residual energy of $O(l^2)$. This continuous case can actually be regarded as a string discretized to infinitely many small pieces. I define the residual energy as the difference of the potential energy between the current discretized one and the continuous one.

Subdividing the spring will bring the entire system closer to the actual continuous case (since the newly introduced vertices will fall down), so the residual energy will decrease. The spring system will start to bounce around upon discretization and I assume that there are damping forces in the system. After discretizing the spring into c pieces of equal length, the new system will have a residual energy of $O(l^2/c)$ when reaching the equilibrium state in the new discretization setting. If the system is in the critical damping condition, the energy will decrease by a factor of e after $t = \sqrt{m_s/\xi}$ seconds, where m_s is the mass of the spring and ξ is the stiffness. Therefore, the recovery time needed from the coarse level to the fine one is $O(\sqrt{m_s/\xi}\ln c)$.

In my case, I have $K = c^{O(1)}$ which depends on the embedded simulator and the collision state. Also I set $\ln K \leq 7$ to cover most of the cases. I use the density and the Frobenius norm of the stretching and bending stiffness matrix in [93] to estimate $\sqrt{m_s/\xi}$. m_s typically ranges from 0.1 to 1, while the value of ξ is between 10 and 100.

Combining all of them above, I have an estimation of $c_0\sqrt{m_s/\xi}/\Delta t$ as the number of iteration steps needed, where m_s is the density and ξ is the Frobenius

norm of the stretching and bending stiffness matrix in [93]. I use $c_0 = 10$ across all of my experiments. In practice, the iteration can also end when no large difference is detected between current and previous results. I found that using my estimation number the difference threshold can be as small as 10^{-3} relative to the scale of the cloth.

In each of the temporal partition, I add an extra simulation steps of $c_0\sqrt{m_s/\xi}/\Delta t$ to refine the starting state, so the total ideal performance gain due to parallelization is

$$\eta = \frac{N}{c_0\sqrt{m_s/\xi}/\Delta t + N/\eta_2} \tag{5.29}$$

Given a cloth material configuration with fixed m_s and ξ, η will have an upper-bound of η_2 if $c_0\sqrt{m_s/\xi}/\Delta t \ll KN/\eta_2$. This can be easily satisfied since the duration $N\Delta t$ is usually from a few seconds to many minutes, and $c_0\sqrt{m_s/\xi}$ is usually smaller than 1.

5.5.5 Implementation Details

There are some minor details in the implementation of the approach. When I take a larger step in the low-resolution simulation, I estimate the change of the state in the corresponding high-res step $u(\mathbf{X}_k^C) - u(\mathbf{X}_{k-1}^C)$ by linearly interpolating the states in between. The same method is also used in the adaptive partitioning method described in Sec. 5.4.2. The recovery iterations also count into the estimation of the current $\tilde{K}$, but do not count into the total number of steps, N, since there is no corresponding step in the low-res level and each processor has the same number of

extra simulation steps, so the system still remains balanced. I regard $\tilde{K}$ as $+\infty$ if the first step of the high-res simulation is not finished at the time I determine n in Sec. 5.4.2. Note that the state $\mathbf{X}$ includes both the position and velocity components. I also refine the velocities in the upsampling phase. When using adaptive remeshing, I obtain the new velocity as the average of the two vertices during edge splitting, following ArcSim [74]. The change of the state is also computed correspondingly.

5.5.6 State Inconsistency

In the extreme cases where the high-resolution mesh is much finer than the low-res one, e.g. 1M versus 100, the shape of the cloth in that case is largely determined by the aggregated effect from details not captured by low-res simulation. Therefore, I cannot recover the exact detail as in the serially simulated one at the transition point, which is referred to as the 'state inconsistency problem'. Enforcing the high-res mesh to match the low-res one using the tracking solver [171] can effectively avoid this problem. So, it can lead the simulation result to follow the movement of low-res one instead, which limits this approach from accuracy-demanding usage in those extreme cases. However, for other usage such as rapid design prototyping, where environmental constraints are mild and K is reasonable, motion difference between two levels is small and I can indeed achieve visually plausible results with high speedup, which are shown in Fig. 5.10 and 5.11. Alternative methods to improve the speedup without harming the accuracy is also discussed later in Sec.5.6.5.

5.6 Results

My method is tested on a large computing cluster with 526 compute nodes, each with 12-core (dual socket), 2.93 GHz Intel processors, 12M L3 cache (Model X5670), and 48 GB memory at 2:1 ratio IB interconnect, MPI for communication. I run one process in each of the cores (compactly assigned). I use up to 128 cores of this cluster to show the linear scalability provided by my theory and up to 512 cores to show the maximum possible speedup in large distributed systems. I could not test on a larger number of cores due to a core limit of 512 per job locally. I use the upsampling function by [74] throughout all of my experiments except in Table 5.4, which uses linearly-interpolated subdivision for fast error computation. As stated in Sec. 5.5.6, this method cannot guarantee the same accuracy as full simulation, which often cannot guarantee the same accuracy as the physical systems. The objective of this work is to generate visually plausible simulation to provide rapid visual feedback for interactive applications, such as rapid design prototyping.

5.6.1 Parameter and Scenario Setting

As mentioned in Sec. 5.4.3, I control the general coarse-to-fine ratio by limiting the smallest mesh size and enlarging the time step of the low-res simulation. Specifically in all of my test cases, the smallest length size of the triangle in the low-res simulation is about 5 times as large as that in the high-res one. The number of iterations in each of the smoothing processes is set to be the same as that in Sec. 5.5.4. I use ARCSim [74] as my base simulator, since it naturally supports

adaptive mesh refinement with an efficient remeshing algorithm. My method can be used in other CPU-based simulators using uniform meshes as well, as long as the upsampling algorithm is specified or implemented. All listed K in the following tables are averaged values across the entire simulation. I show scaling results using figures for clarity.

I use 7 different benchmarks to test the performance and the animation quality of my method: **Blue Dress and Yellow Dress** (Fig. 5.11(a,b)), **Sphere** (Fig. 5.11(c)), **Falling** (Fig. 5.11(d)), **Karate** (Fig. 5.1), **Twisting** (Fig. 5.10(a)) and **Funnel** (Fig. 5.10(b)). The default setting is 20 second simulation at the low-resolution time step of 0.02 sec using 128 cores. I extend the duration to 80 seconds and decrease the time step to make comparisons and validate my theoretical analysis on performance gain. Below are descriptions of each benchmark data.

To the best of my knowledge, previous works did not provide any code or experimental data to public, so the best known practice is to use the reported 'speedup data' in other works with similar scenarios, to minimize the difference due to computing platforms or implementation. I use the timing data of 'Two Cloths Draped' scenario from [152] since it has similar settings as mine (cloth-object interaction), similarly with other benchmarks.

Scenario	Blue Dress	Yellow Dress	Sphere	Falling	Karate	Twisting	Funnel
Original size speedup	74.1	75.0	102	116	96.4	92.3	93.8
4x large size speedup	99.6	109	178	119	103	101	108

Table 5.2: Results on a higher-resolution mesh. I run my system on meshes of higher resolution. Values in the table are the corresponding speedup.

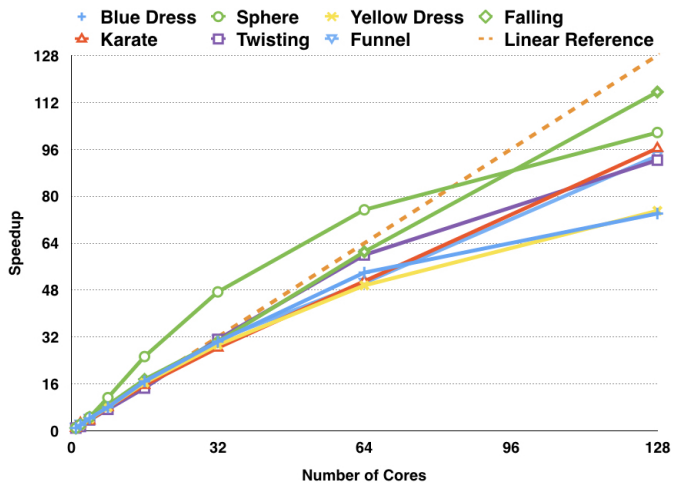


Figure 5.5: Performance scaling result with large low-res time step. A nearly linear scalability is achieved.

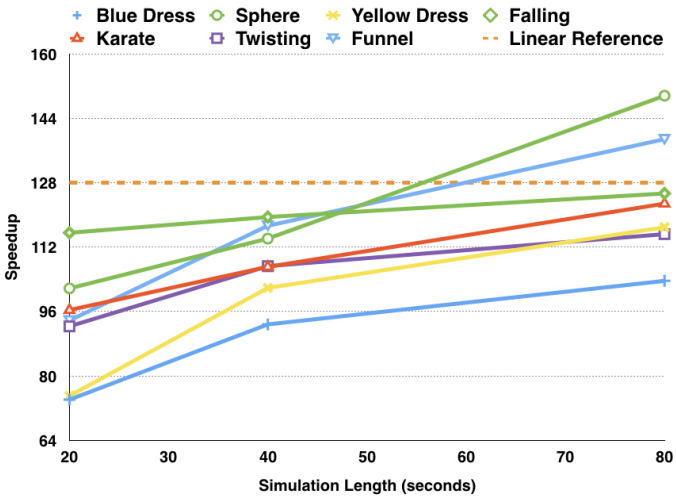


Figure 5.6: Results with increasing length of the simulation. A larger speedup is observed with longer duration of simulation.

5.6.2 Performance

Nearly linear scalability w.r.t. the number of cores. As indicated in Fig. 5.5, my method achieves a good scalability with an increasing number of processors. The reason of the super-linear speedup in the 'Sphere' scene is that it contains rapidly changing contacts with obstacles. When the cloth is free from contact after the sphere passes through, the remeshing algorithm of ARCSim failed

to simplify the mesh effectively, spending an unnecessarily large amount of time simulating simple flat cloth. However, due to the nature of my two-level structure, I maintain a reasonably small number of mesh elements while preserving the quality, and therefore outperform the serial approach significantly. I tested my method on a higher-resolution mesh and observed an even better speedup (Table 5.2) due to the same reason.

Improved scalability with increasing simulation duration. I show in Fig 5.6 that the scalability of my parallelized cloth simulation improves as the duration of the simulation increases. Although the averaging effect of the remaining load imbalance may partially account for it, the most likely reason is from Eqn. 5.29. I have relatively small speedup in 128-core parallelization when simulating a 20-second simulation because the iterative detail recovery algorithm consumes a relatively large amount of time according to Eqn. 5.29. Since the overhead is not dependent on the duration of the simulation and my method is a time-domain parallelization technique, the performance gain improves as the length of the simulation increases due to a smaller portion of the overhead.

Cores	8	16	32	64	128
Uniform partition runtime(s)	5533	3010	1042	684	631
Adaptive partition runtime(s)	4721	2568	928	565	532
Speedup (%)	117	117	112	121	119

Table 5.3: Comparison between different partition schemes. Values in the table are simulation runtime in seconds.

Performance impact on different choices of parameters. To verify my scalability analysis in Sec. 5.4.3 and 5.5.4, I further ran my benchmark with much smaller time steps in low-res simulation. As mentioned in Sec. 5.4.3, increasing

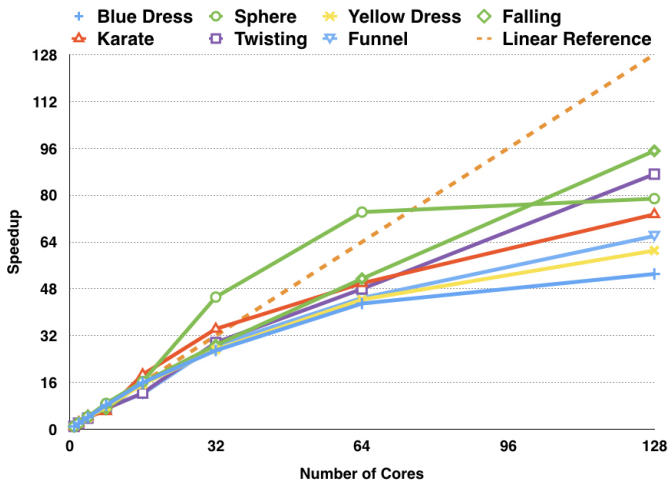


Figure 5.7: Performance scaling result with small low-res time steps. Compared to Fig. 5.5, the speedup for cases with core number larger than 32 is decreased, due to the smaller time steps for low-res simulation.

low-resolution time step is one of the ways to increase the ratio of high-to-low-res simulation time, K. Fig. 5.7 shows that smaller time steps in low-res simulation leads to a sub-linear scaling in all datasets, starting from the 64-core configuration. Although the 'Sphere' dataset has a bigger K due to its simplicity, the scalability starts to degrade at 128 cores as well. The speedup still increases with the simulation duration. However, as it is more closely bounded by K, the gain factor is not as significant as that with large time steps. In practice, a large time step in low-resolution simulations is beneficial to the parallelization performance, but it is limited by (a) the embedded simulation method, (b) the duration of a single frame, and (c) the desired animation quality.

Performance impact on different partition schemes. Table 5.3 shows that by using my adaptive partitioning scheme, I achieve an average of about 120% speedup compared to the uniform partitioned one with the best chosen parameter. In cases such as rapid design prototyping, where the cloth is in continuous contact with

Scenario	Blue Dress	Yellow Dress	Sphere	Falling	Karate	Twisting	Funnel
Time step(low-res)	1/200s				1/100s	1/50s	1/125s
Time step(high-res)	1/200s						1/500s
# of faces(low-res)	5K	6K	8K	6K	4K	4K	4K
# of faces(high-res)	80K	95K	131K	94K	58K	65K	65K
# of triangles(obstacle)	20K	20K	1280	15K	28K	762	4K
K	165	170	172	60	99	188	794
Low-res speed (serial 1-core)	0.6	0.79	1	1.2	0.83	0.22	0.32
High-res speed(OpenMP 12-core)	32.2	44.3	55.9	23.2	27.6	13.7	86.7
My method	0.89	1.14	1.3	1.5	0.91	0.41	1.22
Error before detail recovery	11%	12%	3.2%	22%	29%	46%	16%
Error after detail recovery	4%	6%	0.6%	5%	9%	14%	7%

Table 5.4: Results in the extreme case. I use 512 cores to simulate these scenes. Values in the table are in seconds per frame. The error metric is relative curvature difference compared to serial results in percentage. I use linear interpolated subdivision for fast error comparison.

obstacles, the parameter K remains relatively stable. However, it is still difficult to compute K before simulation begins, since it depends on the specific mesh and collision structure. Furthermore, it is best not to compute the parameter using the first few frames, since the cloth at the beginning can be under constrained without sufficient contact with the obstacles. My adaptive partitioning method here serves as an on-the-fly parameter estimation algorithm in order to achieve good workload balance.

Low-res speed with high-res mesh on a large distributed system. I further test my method in extreme cases where K is relatively small compared to p, which is possible in practice when the computational resources are sufficient. The runtime result is shown in Table 5.4. Although I cannot achieve a speedup as high as 512 due to the limitation of K, I have actually met the upper bound. The serial low-resolution simulation has consumed most of the time so there is very little space to improve in my scheme.

Comparison with previous CPU parallelization work. I compare the performance of my method against other CPU parallelization techniques. Fig. 5.8 shows

that in smaller-scale systems (less than 16 cores), my method can maintain a linear speedup with respect to the single-core system, scaling better compared to previous CPU-based methods using spatial-domain partitioning, e.g. 11x over 16 cores by [154]. For larger-scale systems (Fig. 5.9), I achieved about 50% more efficiency than previous methods such as [152]. In these methods, the processors need to send the information to each other, typically several times, when solving the linear system, resulting in large communication overhead and limited scalability. In contrast, my method only needs to share the states from low-resolution simulations once. Therefore, my method can achieve greater scalability and efficiency in comparison.

In addition, I compare my method with the original embedded OpenMP version of ARCSim. Although a maximum of 2.69x is observed using OpenMP with 2 cores due to a better cache usage in the linear solver, the performance scaling is poor when adding more cores, which results from that the simulation algorithm does not parallelize the remeshing process due to memory access issues. My method disables the OpenMP feature in the ARCSim. Since I parallelize the simulation in time domain, I can avoid memory access control problems, thereby achieving a better speedup.

Method	Speedup over sequential ArcSim [74]	Supports Adaptive Mesh?
Tang et al. [153]	47-58x	No
My method(64-core)	50-75x	Yes
My method(128-core)	75-115x	Yes
My method(512-core)	91-214x	Yes

Table 5.5: **Comparison with GPU method [153].** Other than the scalable speedup gain with more cores, the method is able to naturally support adaptive mesh during the simulation.

Comparison with GPU-based parallelization. Using similar benchmarks as [153],

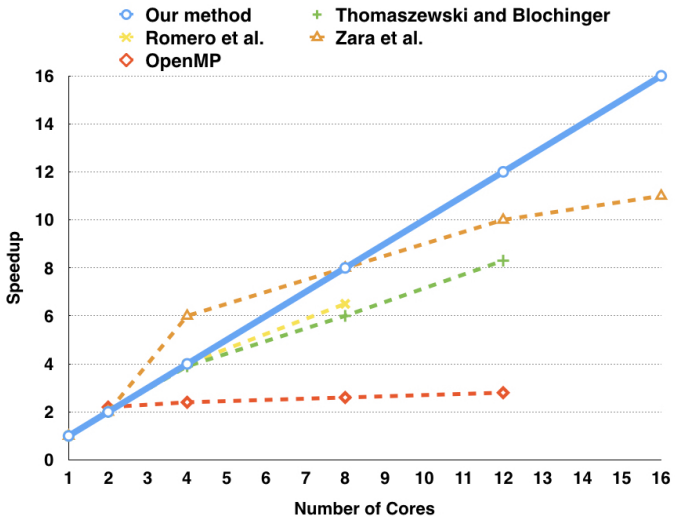


Figure 5.8: Small scale parallelization comparison. My method (in blue solid line) achieves a linear speedup, while others are limited by the communication overhead due to spatial domain partitioning.

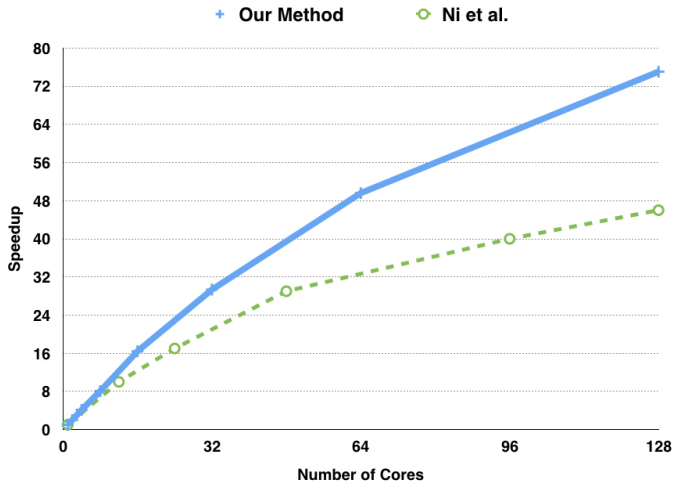
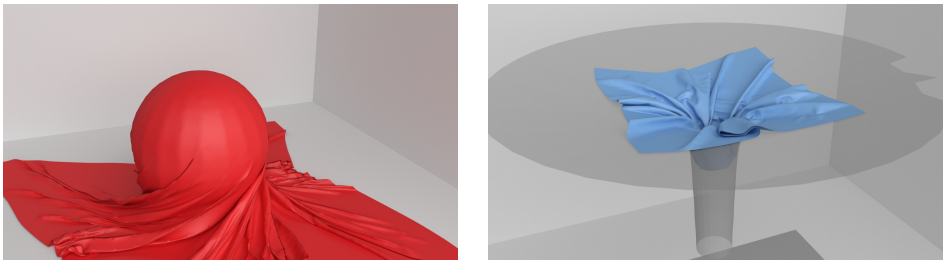


Figure 5.9: Large scale parallelization comparison. My method (in blue solid line) achieves about 50% higher efficiency than [152] using dynamic workload balancing.

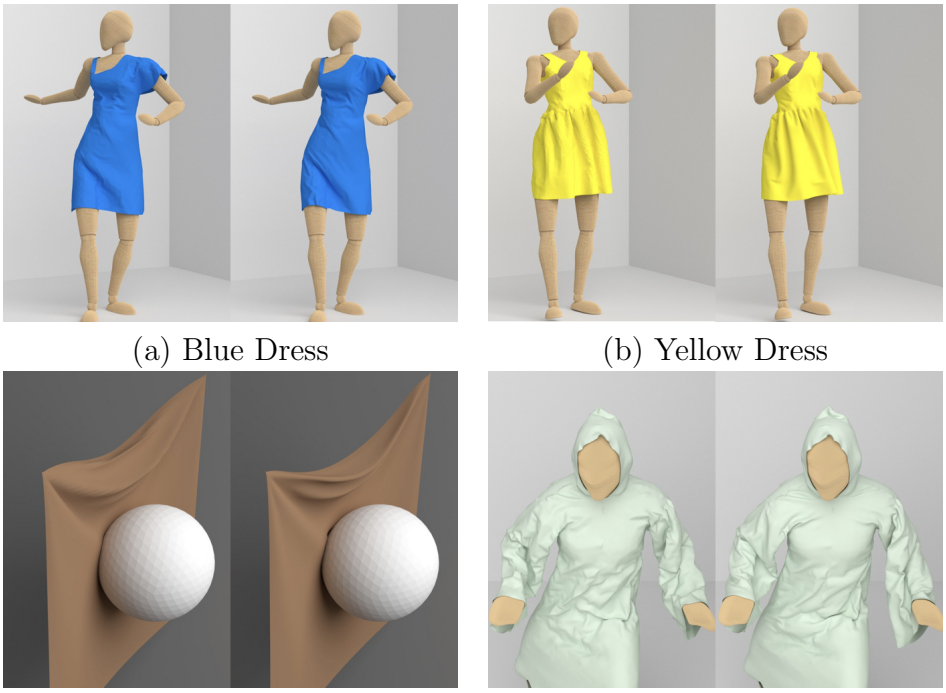
the speedup of my method in a 64-core system configuration is up to 54x in practical scenarios compared to the original ARCSim implementation on a single-core system and achieves a performance gain comparable to the GPU parallelization of [153] (Table. 5.5). However, my method has other distinctive strengths compared to the GPU method. Mine is the first work that can couple an adaptive mesh of varying dimensions during the simulation. I use the same number of triangles for



(a) Twisting (b) Funnel

Figure 5.10: More simulation results (best view with zoom-in in PDF). I have achieved visually plausible and smooth results even in challenging cases involving frequent contacts.

performance comparison, but in practice I can produce similar visual granularity with much fewer triangles using adaptive mesh [74], thereby making my method even faster. Moreover, my performance can be further improved using more cores and a longer simulation sequence, as shown in Fig 5.5 and 5.6.



(a) Blue Dress (b) Yellow Dress

(c) Sphere (d) Falling

Figure 5.11: Refining results (best view with zoom-in in PDF). The left image in each of the example is the upsampled mesh **without** detail recovery, which lacks high frequency details and causes 'popping' artifacts. The right one is the corresponding mesh **using my method**.

5.6.3 Smoothness

Fig. 5.11 and Table 5.4 shows the results before and after the refining algorithm is applied. If directly using the results from the upsampling algorithm, the detail of the cloth is significantly different from the correct one and therefore introduces popping artifacts. After applying the iterative smoothing algorithm, the high frequency information is recovered. I use average curvature distance defined in Eqn. 5.30 to measure the error between the recovered mesh and the original, high-res one simulated using ARCSim on a single core.

$$E = \frac{\sum_{f_1,f_2 \in F} |curv(f_1, f_2) - curv(\tilde{f}_1, \tilde{f}_2)|}{\sum_{f_1,f_2 \in F} |curv(f_1, f_2)|} \tag{5.30}$$

where f_1, f_2 are two adjacent faces in the original mesh, and $\tilde{f}_1, \tilde{f}_2$ are two corresponding faces in my simulation result. I disable remeshing and use linearly-interpolated subdivision for fast comparison. A larger value of the curvature error indicates a sharper edge in the corresponding position and thus a potential artifact. Before my recovery method, a relative error up to 46% is observed, which can cause large 'popping' artifacts in the result animation (Fig. 5.11). By using my technique, the error has decreased by 2-5 times, which is a significant improvement.

5.6.4 Memory and Render Latency

The extra memory footprint introduced by my method is small compared to the high-res mesh. In my experiments, the low-res mesh storage is 5.5% of the high-

res one. I do not render the low-res simulation in my method, and it actually starts at the same time with the first partition of the high-res one. Therefore, my method does not introduce any latency compared to the full-res simulation. In fact, I have achieved a 'pre-fetch' effect for the subsequent partitions due to the very fast, low-res simulation, thereby reducing any potential latency introduced by non-real-time simulation.

5.6.5 Limitations

There are some limitations with this method. First of all, the performance gain is bounded by the ratio of low- to high-resolution simulation time. Other than accelerating the simulation through parallelization in the temporal domain, I can additionally employ GPU implementation to further improve the overall gain. With a factor of $50x$ speedup from GPU [153] and a sufficient number of processors to parallelize the high-resolution simulation, it is possible to accelerate the performance even further. Secondly, the runtime of my method is bounded by a single-step high-resolution simulation time. This implies that at least one simulation step must take place in order to see the result. However, my method accelerates the overall performance, so I can actually achieve 'pseudo-interactivity', where the user can have a very fast visual feedback in parallel. Another possible direction is to implement a hybrid domain decomposition scheme, allocating some processors for spatial-domain parallelization to accelerate the single-step runtime. My approach provides plausible visual results in practical real-time applications, like rapid design prototyping.

However, as stated in Sec. 5.5.6, This approach may not be suitable in applications requiring high precision. In practice, the resulting cloth can sometimes appear slightly stiffer than the original one.

5.7 Conclusion and Future Work

In this chapter, I introduce a novel temporal-domain parallelization method for practical cloth simulation such as rapid design prototyping. Taking the advantage of faster simulations on coarser meshes, I parallelize the cloth simulation in time with accelerated computation and minimal communication overhead. I also proposed an iterative detail recovery algorithm to minimize the visual artifacts due to the state transitioning from coarse to fine meshes. My method outperforms existing CPU- and GPU-based parallelization techniques on a diverse set of benchmarks. It offers high efficiency and nearly linear scalability on large distributed systems, while maintaining high-fidelity visual simulation of the cloth. The scalability of my method is dependent on the ratio of low- to high-resolution simulation time, the length of the simulation, and persistence of contacts with obstacles. Since this method utilizes only time-domain parallelization, a natural extension would be a hybrid decomposition scheme that may provide a potential usage in short-duration simulation or in circumstances with memory constraint.

This work has been published in the proceedings of ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA) 2018.

Chapter 6: Dynamics-Inspired Garment Draping Prediction

6.1 Introduction

Other than accelerated cloth simulation proposed in Chapter 5, one can also use deep learning to directly predict the garment draping given the body as input. This approach offers real-time feedback to the user guaranteed by GPU-based network inference. Within the training distribution, learning-based methods can also produce draping results as good as simulated ones, accurately revealing information, such as fitting and material perception.

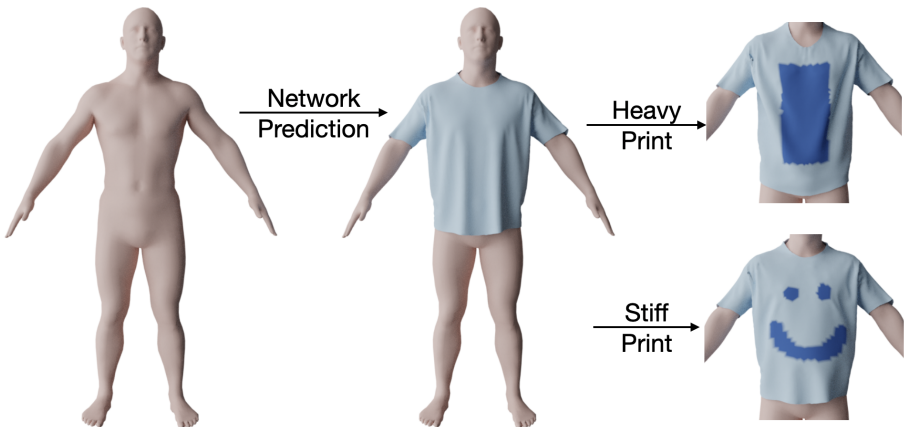


Figure 6.1: My model learns how to drape garments in two stages: first supervising the network with a physics-inspired loss, then optimizing the output according to the physics of specified garments, such as new print shapes or materials, in a self-correcting way.

However, there exists a number of major challenges for interactive learning-

based virtual try-on. First, it is well-known that machine learning models tend to produce overly smoothed results when using per-vertex distances as their main loss [11, 180, 181]. Although previous works [11] addressed this issue to some extent, they are limited to a narrow set of shapes composed of nine distinct bodies. Next, while garments are often composed of different materials (*e.g.* the frontal graphics print in T-shirts), existing works typically model a single one because it is impractical to consider all combinations of different graphics prints shapes at training time. Finally, it is expected that the garments predicted by the network model is fit-accurate, although the most common losses in related work (per-vertex geometry errors) do not necessarily entail fit accuracy. This translates into changes in the overall shape of the garment and violations to its material properties.

In this chapter, I propose a novel semi-supervised framework to address all issues above. One key idea is that physical constraints can be reformulated as geometric loss functions that can be optimized during training. However, using the exact physical loss functions from scratch does not result in good draping due to their complexity. Therefore, I first train my model with supervised, physics-inspired loss functions, and then optimize the model output individually for each sample to conserve the actual physical energy and minimize/eliminate geometric intersections. Given their superior quality compared to the initial predictions, the resulting optimized samples can then be re-used to further improve the initial feed-forward network.

Overall, the key contributions of this work include:

- A semi-supervised framework that enables easy integration of constraints into the deep learning model.
- Introduction of novel loss functions that encode geometric, physical, design, and tailoring constraints.
- A novel encoder/decoder network that effectively captures global and local features from the input and dynamically aggregates neighborhood information.
- A new self-correcting method based on data augmentation that enables both more accurate predictions and reduced data preparation time.

6.2 Related Work

Drape prediction systems fall mainly in three categories: physics-based simulation, learning-based garment generation, and direct estimation of garments from real-world.

Physics-based Cloth Simulation. Physics-based garment simulation systems usually include spatial discretization [27, 91] and different forms of simulations [24, 182]. Although several techniques have been proposed to speed up cloth simulation, including GPU acceleration [162, 183], spatial and temporal parallelization [152, 170, 184, 185], and other techniques [186, 187, 188], real-time, physically accurate cloth dynamics for any given human shape remains illusive [74].

To reduce computation time, several works produce high frequency wrinkles on low-resolution meshes as a practical trade-off [173, 178, 186, 189, 190], but are

typically limited to tight garments. In tight garments, cloth material and overall drape have a negligible effect in the wrinkles. In contrast, my feed-forward model can achieve real-time performance even when generating high-resolution garments. Inspired from optimization-based simulation [191], I use the physics-based metrics as some of the loss functions, resulting in realistic folds and wrinkles.

Learning-based Garment Draping. As a faster alternative to simulation, learning-based approaches have been developed for draping garments with better visual quality and realism, including normal map generation [124], KNN body-garment fusion [8], wrinkle-assisted design [9], displacement regression [192, 193], motion-conditioned auto-encoder [194], and least-square approximation [195]. However, these works are limited in different aspects: [124] do not provide geometric details; [8, 193, 195] generate relatively smoothed results; [193, 194, 195] do not generalize to a wide range of body shapes; [9] requires user knowledge of wrinkle formation; and [192] cannot deal with loose clothing and produces a single mesh containing both body and garment fused. In contrast, my method takes only a human body mesh as input and directly regresses the garment mesh as output with realistic geometric details.

Other works tackle the same task as my system does. Santesteban *et al*. [180] used RNN to capture the motion-dependent garment shapes on the body. Vidaurre *et al*. [181] extended the approach to cover more size variations of different design parts. Since [180, 181] supervise their methods with a pure per-vertex distance loss, their model suffers from smoothed folds especially when the garment is loose. To

remedy the smoothing effect, Patel *et al*. [11] interpolated high frequency results from different anchors based on the inverse of per-vertex distances between garment predictions on the canonical pose. Despite its success, it is based on a limited discrete set of training shapes, which can pose generalization problems as can be observed in Sec. 6.5.6. In contrast to the three methods above, I design an exhaustive set of loss functions inspired by physics constraints, including the minimization of differences in the spectral domain.

Garment Capture and Estimation. Instead of simulating garments, some systems capture garments from real-world data. These systems focus on inferring the mesh geometry using visual features and retargeting it to the desired body. State-of-the-art methods have varying types of input, including images [12, 68, 110, 116, 120, 121, 196], mesh scans [22, 197], RGB videos [2, 198], or videos together with the depth channel [122, 199]. Despite the appeal of avoiding altogether physical simulation in both train and test stages, capturing methods still have a number of drawbacks. First, it is hard to collect a dataset large enough to train a model that can cover the large variation of human shapes. Moreover, simple retargeting often fails to account for non-rigid and non-linear transitions of the garment deformation between different bodies. Note that there is an intrinsic difference between my method and capture systems: while my method focuses on generating drapes *given just the underlying body*, capture methods extract the garment from existing media.

6.3 Method

In this work, I focus on a wide range of human body sizes rather than different poses. To this end, I used the SMPL model [30] to generate a set of 20,000 bodies of varied male shapes (see supp. mat. for the equivalent female model) following a uniform distribution of BMIs between 15 and 65, all with the same A-pose. In comparison, my analysis indicates that from the 9 discrete bodies used by TailorNet [11] only one has a BMI over 30. Given that 42.4% of the population in the US has obesity (BMI $>$ 30) [200], TailorNet does not have enough coverage on large bodies.

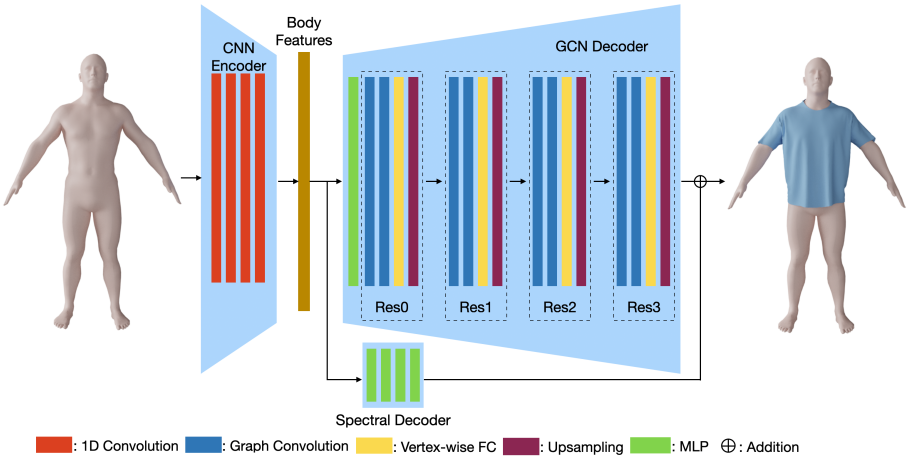


Figure 6.2: Overall structure of my network. I use 1D CNN as my encoder and Graph Convolutional Network (GCN) as the decoder with four different resolutions. I additionally employ an MLP-based spectral decoder for high frequency preservation.

The overall structure of my network is shown in Fig. 6.2. I first pass the ordered vertex coordinates of the body to a 1D CNN network to extract features (Sec. 6.3.1). Next, I transform the features corresponding to vertices in the body using 1-layer MLP, and distribute them to the vertices of the garment according to

fixed correspondences between body and garment. I design a resolution hierarchy for the Graph Convolutional Network (GCN) decoder to capture both global and local information (Sec. 6.3.2). To improve the higher frequency wrinkles, I finally introduce an MLP branch that predicts the garment residuals in the spectral domain (Sec. 6.3.3). The losses used for training this network are described in Sec. 6.3.4.

6.3.1 Encoder

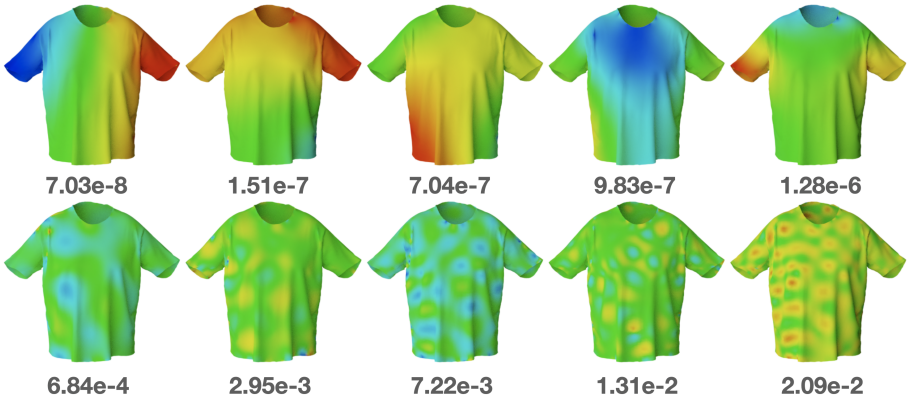


Figure 6.3: Visualization of the eigenvectors and their eigenvalues. Red denotes large weights, blue small ones.

Unlike PointNet [99], which obviates the vertices order by spatially pooling its features, I would like to exploit the fixed topology of the input SMPL vertices. Since the input mesh does not have a neighbor structure constant across vertices, a natural choice for this would be a GCN. However, in Sec. 6.5.2 I show that my simpler alternative, a 1D CNN, empirically outperforms the GCN. I hypothesize that this might be due to the large amount of model capacity devoted to body parts unrelated to the garment (*i.e.* head, hands, etc). In practice, using a 1D CNN on the original SMPL vertex order is more efficient and captures most SMPL neighborhoods, since

91.24% of the SMPL vertices have their adjacent indexed neighbors being adjacent in topology, and 94.01% are at most two-hops away.

6.3.2 GCN-Based Decoder

Following a similar reasoning as in the previous section, I would like to capture local relations between the garment vertex neighborhoods. Therefore, I use a graph convolutional network in the decoder. A common graph convolutional layer is defined as:

$$\mathbf{y} = \mathbf{f}_\theta(\mathbf{A}\mathbf{x}) \tag{6.1}$$

where $\mathbf{A}$ is the aggregation matrix which collects and processes the information in the neighborhood in an isotropic way, and $\mathbf{f}_\theta$ is the nonlinear activation function for feature extraction. The expressiveness of this network is inherently limited since the constant aggregation matrix cannot adapt its neighbor aggregation weights. Attention-based GCN [201] addresses this issue by proposing an MLP to estimate the aggregation parameters given the vertex features:

$$\mathbf{y} = \mathbf{f}_{\theta_1}(\mathbf{A}_{\theta_2}\mathbf{x}) \tag{6.2}$$

$$\mathbf{A}_{\theta_2}[i,j] = MLP(\mathbf{x}_i, \mathbf{x}_j) \tag{6.3}$$

In contrast, I propose to learn the aggregation parameters independently per vertex, without an explicit dependence on the features:

$$\mathbf{A}_{\theta_2}[i,j] = \theta_2[i,j] \tag{6.4}$$

Another particularity of my decoder is its hierarchical nature. Analogous to up-scaling in 2D image generation, feeding the encoded features to a coarsened mesh helps distributing global features to broader regions of the garment. To linearly upsample the features, I use the barycentric weights of the corresponding higher resolution vertices with respect to the lower resolution ones, all in UV space. I use four resolutions in the decoder for all experiments.

6.3.3 Spectral Domain Decomposition

Simulation systems are known to be input-sensitive; negligible differences in the input or initialization can result in substantial differences in the outputs, specially in the high frequency domain. Supervision on vertex positions tends to average those multiple possible outcomes, smoothing out its predictions. However, the high frequency content of the garment is critical for garment perception, since it is highly correlated to garment materials and tightness. This motivates the need to inspect the spectral components of the garment mesh [202]. Specifically, I apply the eigen decomposition on the Laplacian operator:

$$\mathbf{L} = \mathbf{U}\mathbf{D}\mathbf{U}^{-1} \tag{6.5}$$

where $\mathbf{U} \in \mathbb{R}^{n*n}$ and $\mathbf{D}$ are the eigenvectors and the diagonal matrix of the eigenvalues. I pick the subset of eigenvectors $\mathbf{V} \in \mathbb{R}^{n*k}$ corresponding to the smallest k eigenvalues. The spectral coefficients of a mesh $\mathbf{c} = \mathbf{V}^\top \mathbf{x}$ thereby represent the mesh components with lowest impact on Laplacian values. This method rejects the highest frequencies (typically noisy) since high frequency entails large local changes, which have a large impact in the Laplacian (and therefore large eigenvalues). The visualization of the eigenvectors is shown in Fig. 6.3.

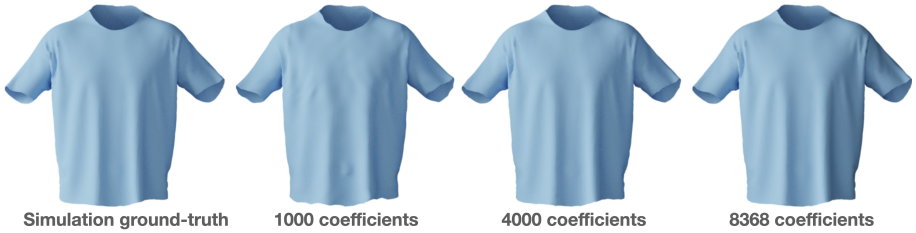


Figure 6.4: Reconstructions for different numbers of coefficients.

This spectral decomposition is used by introducing an MLP-based branch in the decoder network to account for residuals of the spectral components. I output the coefficients $\Delta\mathbf{c}$ of the 4,000 eigenvectors with the smallest eigenvalues, which are sufficient for reconstruction (see Fig. 6.4). These coefficients are then transferred back to the spatial domain $\Delta\mathbf{x} = \mathbf{V}\Delta\mathbf{c}$ and added to the final result.

I also introduce a spectral loss during training. This loss ensures that high frequency components, which typically result in small vertex displacements, deserve proper attention in the supervision of my model.

6.3.4 Loss Functions

I design a number of loss functions to supervise and guide the model towards realistic outputs. First of all, I split the output into a correspondence point set and a displacement map:

$$\mathbf{x} = \mathbf{c}_x + \mathbf{d}_x \quad \mathbf{y} = \mathbf{c}_y + \mathbf{d}_y \tag{6.6}$$

where $\mathbf{c}$ are the correspondence (closest) points on the body surface and $\mathbf{d}$ are the displacements w.r.t. the correspondence, and $*_x$ represents the prediction while $*_y$ represents the ground-truth. This partition of garment into body plus displacement is common in the literature [11, 196], but here it enables an important direction loss that prevents intersections and preserves normals:

$$\mathcal{L}_{dir} = R(-\frac{\mathbf{n}^\top(\mathbf{c}_x - \mathbf{c}_y)}{\|\mathbf{c}_x - \mathbf{c}_y\|}) + (1 - \frac{(\mathbf{x} - \mathbf{c}_y)^\top \mathbf{d}_y}{\|\mathbf{x} - \mathbf{c}_y\|\|\mathbf{d}_y\|}) \tag{6.7}$$

where R denotes relu, and $\mathbf{n}$ is the normal direction at $\mathbf{c}_y$. The first part of the direction loss constrains the correspondence to be outside of the body, while the second part constrains the direction of the prediction to be similar to the ground-truth. Since $\mathbf{c}_y$ is defined as the closest point on the body surface to the garment vertex $\mathbf{x}$, minimizing the direction loss can help generate results with fewer intersections and better normal consistency.

I also use per-vertex L1 difference of these two components separately to su-

pervise the overall shape:

$$\mathcal{L}_{V2V} = \|\mathbf{c}_x - \mathbf{c}_y\|_1 + \|\mathbf{d}_x - \mathbf{d}_y\|_1 \tag{6.8}$$

Additionally, I applied physics-based losses to learn the correct deformation of the garment. The key idea here is to transfer the physics constraints applied in simulation to geometric differences. I choose two aspects that reflect the physics: edge lengths and deformation gradients. First, the edge loss measures the difference of the edge lengths relative to the ground-truth:

$$\mathcal{L}_e = \frac{1}{|E|} \sum_{(\mathbf{u},\mathbf{v})\in E} \frac{|\|\mathbf{u}_x - \mathbf{v}_x\| - \|\mathbf{u}_y - \mathbf{v}_y\||}{\|\mathbf{u}_y - \mathbf{v}_y\|} \tag{6.9}$$

where $*_x$ are the predictions and $*_y$ are the ground-truths, and E is the edge set of the mesh. This loss guides the model to generate more wrinkles because smoothed results often have smaller overall circumference (thereby larger $\mathcal{L}_{edge}$) than ground-truths. Additionally, I define a loss that supervises on the difference of the deformation gradient of each face in the mesh:

$$\mathcal{L}_d = \sum_{\mathbf{f}\in\mathcal{M}} \|\mathbf{F}_x(\mathbf{f}) - \mathbf{F}_y(\mathbf{f})\|_1 \tag{6.10}$$

$$\mathbf{F}_x(\mathbf{f}) = \mathbf{x}(\mathbf{f})\mathbf{X}^{-1}(\mathbf{f}) \tag{6.11}$$

where $\mathbf{F}(\mathbf{f})$ is the deformation gradient [203] of face $\mathbf{f}$ in the mesh $\mathcal{M}$, defined as the change of the world-space coordinates ($\mathbf{x}$) per unit change of the material-space

coordinates ($\mathbf{X}$) within each triangle. This loss is less intuitive than the edge loss, but provides better alignment to the ground-truth regarding the potential energy and the internal forces it generates. For example, the deformation gradient can represent the shear stress and bulk stress separately, while the edge-based loss cannot.

To capture the curvature and higher frequency errors, I further define a Laplacian Difference loss, and a spectral loss as described in Sec. 6.3.3:

$$\mathcal{L}_l = \sum_{k=0}^{3} \|\mathbf{L}_k(\mathbf{x} - \mathbf{y})\|_1 \tag{6.12}$$

$$\mathcal{L}_s = \|\mathbf{V}^\top(\mathbf{x} - \mathbf{y})\|_1 \tag{6.13}$$

where $\mathbf{L}_k$ is the Laplacian operator on the mesh graph at the k-th resolution, and $\mathbf{V}$ are the eigenvectors of the Laplacian operator on the original mesh defined in Sec. 6.3.3. I apply the Laplacian difference loss in different resolutions to account for wrinkles and folds of different sizes.

The total loss is the sum of all losses defined above:

$$\mathcal{L} = \mathcal{L}_{V2V} + \mathcal{L}_{dir} + \mathcal{L}_e + \mathcal{L}_d + \mathcal{L}_l + \mathcal{L}_s \tag{6.14}$$

6.4 Physics-Enforced Optimization

During inference on unseen input bodies, it is likely that the drape prediction does not reach a stable dynamical state because the corresponding potential energy may not be fully minimized in all cases. More importantly, it is not uncommon

in practical try-on applications that the garment is composed of materials different from the ones used in training. For example, a frontal graphic print of a T-shirt is usually stiffer and heavier than the rest of T-shirt. In theory such situations could be solved by training with the appropriate data, but this solution turns impractical when I consider not only a large variety of graphics print shapes for garments, but further more the infinite variety of them in services that allow you to create your own print.

To address the problems above, I propose to optimize the inferred models on specific samples at runtime. I finetune the network weights for each sample to minimize the potential loss of the garment defined below:

$$\mathcal{L}_p = \mathcal{L}_g + \mathcal{L}_{st} + \mathcal{L}_b \tag{6.15}$$

$$\mathcal{L}_g = \sum_{\mathbf{v} \in \mathcal{M}} m(\mathbf{v}) \mathbf{g}^\top \mathbf{x}(\mathbf{v}) \tag{6.16}$$

$$\mathcal{L}_{st} = \sum_{\mathbf{f} \in \mathcal{M}} \mathbf{S}(\mathbf{f}) \quad \mathcal{L}_b = \sum_{\mathbf{e} \in \mathcal{M}} \mathbf{B}(\mathbf{e}) \tag{6.17}$$

where $\mathcal{L}_p$, $\mathcal{L}_g$, $\mathcal{L}_{st}$, and $\mathcal{L}_b$ are the potential loss functions and its components: gravity, stretching, and bending energy, respectively. $\mathcal{M}$ is the predicted mesh, $m(\mathbf{v})$ and $\mathbf{x}(\mathbf{v})$ is the mass and coordinates of vertex $\mathbf{v}$, $\mathbf{S}(\mathbf{f})$ is the stretching energy of face $\mathbf{f}$, and $\mathbf{B}(\mathbf{e})$ is the bending energy of two adjacent faces with common edge $\mathbf{e}$. I follow the definitions of the stretching and bending energy from the simulator I used [74]. In short, material stiffness coefficients are multiplied to elements in the Green Strain of $\mathbf{f}$ and the curvature of $\mathbf{e}$, respectively. To make the optimization

collision-aware, I introduce a penetration loss function [8].

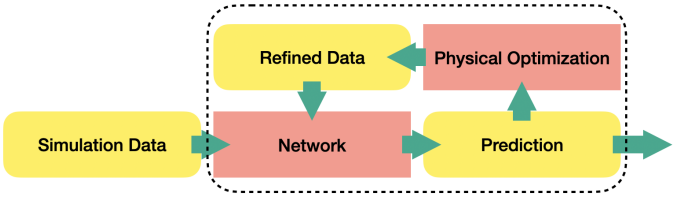


Figure 6.5: The semi-supervised self-correcting training pipeline.

The proposed optimization step serves two purposes in my pipeline. First, it can be used to generate better training data. I empirically show in Table 6.5 that my optimization generates garments with lower potential than the simulated data, even when both are initialized with my network. My hypothesis is that the lack of hard penetration constraints and kinematic limits helps my optimization find a more stable state. Re-training the network with this data can be regarded as self-correcting unsupervised learning since the training data has been produced by the previous network, turning the entire pipeline into a semi-supervised learning framework.

Second, it can be adapted to materials that are not covered by the original model, which covers homogeneous T-shirts. The optimization allows us to predict drapes where the T-shirts contain graphic prints with different shapes and materials (Fig. 6.1). To achieve this, I simply minimize the potential loss of the new system containing the new graphic print. Results in Sec. 6.5 show that the optimization is indeed effective in performing these two tasks.

6.5 Experiments

I conduct several experiments to evaluate my method, including ablation studies testing several different configuration of my system, accuracy tests of the physics optimization, comparisons with previous work, and generalization over multiple T-shirt sizes.

6.5.1 Data Generation

I use the SMPL [30] body model to generate my input dataset. Shape parameters are sampled following a uniform distribution in BMI space with range from 10 to 65 (see Fig. 6.6). In terms of pose, I restrict myself to a fixed A-pose to concentrate on a wide shape variation.

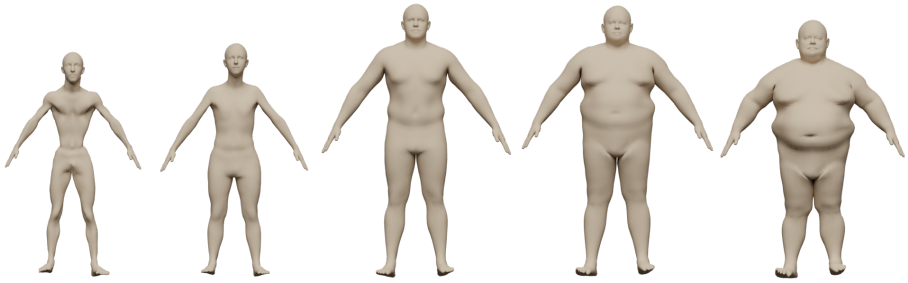


Figure 6.6: Bodies at BMI percentiles 10, 30, 50, 70 and 90%.

I use the T-shirt from TailorNet [11] as my garment template, after retopologizing it to generate a proper UV template for simulation. To generate the ground-truth garments, I initialize the garment to a precomputed configuration coherent with the T-pose average SMPL body. ArcSim [74] is used to simulate the garment movements while the body is linearly morphed into the target shape with A-pose.

The body morphing lasts for two simulation seconds in total, and ArcSim continues to simulate the garment dynamics on the body for two more seconds to obtain stable results. In total, I generate 20,000 such human bodies and garments, where the training/validation/test split is 8:1:1 (for this reason the tables are annotated as *16k* for the full dataset of 20000 samples).

6.5.2 Ablation Study

In this section I evaluate different components of my method in terms of the loss functions defined in Sec. 6.3.4 except from $\mathcal{L}_{V2V}$, which is replaced by the more intuitive mean euclidean distance (ME). I also track the ratio of all garment vertices which are intersected with the human body, p. In the tests in this section, all system components are held constant except for the one being tested.

Encoder. In this experiment my 1D CNN encoder is replaced with other approaches like PointNet [99], the original GCN [204] and my modified GCN (see Sec. 6.3.2), where I replaced upsampling for downsampling operations. Table 6.1 shows that my CNN encoder outperforms other approaches. GCN performs worst because a large part of its capacity is used in feature extraction on body regions of the body irrelevant for my task (*e.g.* hands and feet).

Encoder Type	ME (cm)	$\mathcal{L}_l$ (cm)	$\mathcal{L}_e$ (%)	$\mathcal{L}_s$	$\mathcal{L}_d$	p (%)
PointNet [99]	0.35	**0.16**	9.36	1.1e-3	4.6e-3	**0.05**
GCN [204]	0.45	0.18	10.32	1.2e-3	5.1e-3	0.1
GCN (mine)	0.36	**0.16**	9.54	1.1e-3	4.7e-3	**0.05**
CNN (mine)	**0.33**	**0.16**	**9.21**	**1.0e-3**	**4.5e-3**	**0.05**

Table 6.1: Encoders ablation. ME stands for Mean Euclidean.

Decoder. For comparison, I replaced my multi-scale GCN decoder (see Sec. 6.3.2) with a 2D CNN decoder that represents the garment in UV space [124], an MLP, and the original GCN [204]. My final model achieves the best results in most of the metrics, especially when adding the spectral residual connection. Although the 2D CNN decoder have slightly smaller intersection rates than mine, its large errors in terms of edge lengths and face deformations indicates that the model fails to learn the dynamics of the garment.

Decoder Type	ME (cm)	$\mathcal{L}_l$ (cm)	$\mathcal{L}_e$ (%)	$\mathcal{L}_s$	$\mathcal{L}_d$	p (%)
2D CNN	0.54	0.49	39.45	2.0e-3	22.2e-3	**0.04**
MLP	1.62	0.25	15.71	2.0e-3	7.5e-3	6.96
1D CNN	5.02	0.38	24.51	3.2e-3	11.6e-3	24.67
GCN (original)	1.92	0.41	37.01	3.5e-3	19.8e-3	2.56
GCN (mine)	0.34	**0.16**	9.36	1.1e-3	4.6e-3	0.06
GCN (mine) + spec	**0.33**	**0.16**	**9.21**	**1.0e-3**	**4.5e-3**	0.05

Table 6.2: Decoders ablation. ME stands for Mean Euclidean.

Loss functions. I verify the effectiveness of my loss functions by comparing the test metrics of the learned model when disabling the loss functions one at a time. The main baseline is the per-vertex L2 loss commonly used in previous works [11, 180, 181]. As shown in Table 6.3, all my designed loss functions contribute to smaller overall errors. It can be observed that the intersection ratio is drastically reduced simply by replacing the baseline L2 loss to $\mathcal{L}_{V2V}$ that trains the correspondence and displacement separately (Eqn. 6.8). Introducing the direction loss $\mathcal{L}_{dir}$ further reduces the intersection close to their minimum. The losses related to local shape ($\mathcal{L}_p$), deformations ($\mathcal{L}_e$ and $\mathcal{L}_d$), and spectral energy ($\mathcal{L}_s$) are reduced by introducing the rest of the losses, with negligible negative effects in other metrics.

Loss Functions	ME (cm)	$\mathcal{L}_l$ (cm)	$\mathcal{L}_e$ (%)	$\mathcal{L}_s$	$\mathcal{L}_d$	p (%)
$\mathcal{L}_{base} = L2$	0.4	0.21	12.35	1.2e-3	6.3e-3	2.05
$\mathcal{L}_0 = \mathcal{L}_{V2V}$	**0.33**	0.21	12.93	1.2e-3	6.7e-3	0.41
$\mathcal{L}_1 = \mathcal{L}_0 + \mathcal{L}_{dir}$	0.37	0.23	13.93	1.3e-3	7.2e-3	0.04
$\mathcal{L}_2 = \mathcal{L}_1 + \mathcal{L}_l$	0.36	0.18	11.39	1.2e-3	5.7e-3	**0.03**
$\mathcal{L}_3 = \mathcal{L}_2 + \mathcal{L}_e$	0.35	**0.16**	9.55	1.1e-3	4.7e-3	0.04
$\mathcal{L}_4 = \mathcal{L}_3 + \mathcal{L}_s$	0.34	**0.16**	9.36	1.1e-3	4.6e-3	0.04
$\mathcal{L}_5 = \mathcal{L}_4 + \mathcal{L}_d$ (mine)	**0.33**	**0.16**	**9.21**	**1.0e-3**	**4.5e-3**	0.05

Table 6.3: Losses ablation. ME stands for Mean Euclidean.

6.5.3 Optimization for Semi-Supervision

This section show how my optimization method described in Section 6.4 not only adapts to novel runtime conditions, but also allows us to train a system which achieves high accuracy and fast inference with less data. To validate this, I train my model with 25% amount of the original simulated training data (4,000 samples), and re-generate a full dataset (16,000 samples) where each sample is computed by a 100-iteration optimization from the model prediction. Keeping body input and train/validation/test split exactly the same, I train an additional model based on the new dataset and compare the performance to the model that is trained on the original one.

The test results are shown in Table 6.4. The initial models are denoted as *4k* and *16k* regarding different training data sizes, optimized results are marked with *-opt*, and *4k-retrain* is my retrained model on the optimized data. I evaluate the systems in terms of realism, training and inference time. Realism is evaluated in terms of percentage of intersections and the potential defined in Eqn. 6.15. I use this potential since the original physically simulated garments are not physically optimal and therefore the comparison in terms of per-vertex distance is misleading. The end-to-end training time denotes the total time needed to generate the first sample. For

example, for *4k-retrain* it is the sum of the times from simulation (2,666 hours), *4k* training (3 hours), the optimization of all samples (31 hours), and final retraining (10 hours).

Training with more data results in higher accuracy but longer training times. After optimization, the total potential decreases and remaining intersections are resolved. Although optimization is faster than the simulator, it is prohibitively slow for real time applications. By using the optimized samples for the retrained network, I achieve real-time performance with a fairly competitive physical accuracy. Note that by retraining my system with only 4,000 samples (*4k-retrain*), my end-to-end training time is shorter than that of the original system trained on 16,000 samples.

Models	$\mathcal{L}_p$	p (%)	End-to-end training time (h)	Inference time
Simulation	1.0e-4	0.00	0	40min
4k	8.8e-5	0.1	2,669	17 ms
16k	7.5e-5	0.05	10,677	
4k-opt	4.3e-5	0.00	2,669	7 s
16k-opt	4.1e-5	0.00	10,677	
4k-retrain	5.6e-5	0.04	2,710	17 ms

Table 6.4: Self-correcting pipeline ablation study. Training with more data (*16k* vs. *4k*) results in higher accuracy but longer end-to-end training time due to data generation. My optimization (*-opt*) improves the performance but has long inference time. Re-training on the optimized data (*4k-retrain*) achieves small test errors, short training time and real-time inference.

6.5.4 Optimization for Graphic Print

The optimization method enables the draping adaptation to unseen material designs. In this experiment, I generate two datasets, one with a stiffer and denser graphic print on the T-shirt front and one without. I train two models from the

datasets separately as the baselines, and also run the optimization process on the model trained with blank T-shirts to adapt to the new print. In this experiment I consider the potential $\mathcal{L}_p$ according to the materials of the T-shirt with a frontal graphic print for both optimization and evaluation. I run this process over all the bodies in my test set.

Table 6.5 shows that training with the correct offline data helps reduce minimally the potential, but the optimization achieves a much stronger error reduction. A qualitative example is also shown in Fig. 6.7, where it can be observed that the frontal print forms different wrinkles in the simulation, which are correctly captured in the optimized result. This process enables model generalization at runtime without new data generation, which is crutial if the amount of material configurations is very largely or simply unknown at training time. Note that simulation initialized from the model prediction is not only slower, but also less accurate than my optimized results.

Models	$\mathcal{L}_{st}$	$\mathcal{L}_b$	$\mathcal{L}_g$	$\mathcal{L}_p$	Runtime	$p(\%)$
w/o Print	7.6e-5	4.5e-7	-9.2e-7	7.5e-5	17ms	0.05
w/ Print	7.5e-5	4.4e-7	-9.2e-7	7.4e-5	17ms	0.05
Optimized	4.2e-5	5.1e-7	-7.8e-7	4.1e-5	7s	0.00
Simulated	1.7e-4	6.2e-7	-9.6e-7	1.7e-4	40min	0.00

Table 6.5: Adaptation to new materials. The optimized results is better than the network estimations and the simulation initialized with the model predictions with print, while still being 342x faster.

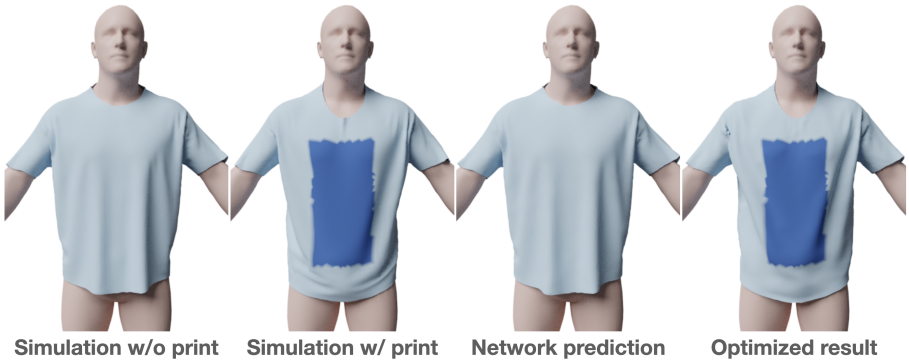


Figure 6.7: Qualitative examples of self-correcting optimization. My method can adapt the model to unseen materials.

6.5.5 Quantitative Comparisons

I compare quantitatively my system (*16k* in Table 6.4) with the system most closely related to mine, TailorNet [11]. Unfortunately I cannot compare to other relevant systems like [180, 181] since they do not provide publicly available code or data; however, a qualitative comparison is provided in Sec. 6.5.6 for completeness. As discussed in Sec. 6.2, remaining works do not fall into the same category as mine, so I do not include them for comparison.

I compare against Tailornet retrained on my own data. Note that for my new dataset, each of the anchor shapes contains a single pose, so learned weighted sum of anchor-based high frequency prediction becomes a learned weighted sum of fixed garments (Table 6.6 *w/ anchor*). To provide more capacity to the high frequency network, I propose an alternative which trains a single high frequency network without anchors (Table 6.6 *w/o anchor*). The original Tailornet could not be used for quantitative comparison since I had to retopologize their template to create a UV map, resulting in a different vertex count. As shown in Table 6.6, my method outperforms TailorNet versions in all test metrics, reducing the errors by

44% ($\mathcal{L}_l$) to 98% (p). I believe the decoder and the loss from TailorNet are the causes of this gap. First, my GCN decoder with learned aggregation outperforms their MLP decoder. Second, I have shown in Sec. 6.5.2 that the per-vertex distance loss from TailorNet is outperformed by my physics-inspired loss.

Methods	ME (cm)	$\mathcal{L}_l$ (cm)	$\mathcal{L}_e$ (%)	$\mathcal{L}_s$	$\mathcal{L}_d$	p (%)
[11] w/ anchor	1.4	0.4	26.59	2.5e-3	14.1e-3	4.89
[11]-post	1.4	0.4	26.4	2.5e-3	14.0e-3	0.03
[11] w/o anchor	1.36	0.29	17.65	2.1e-3	8.8e-3	3.1
Mine	**0.33**	**0.16**	**9.21**	**1.0e-3**	**4.5e-3**	**0.05**
Improvement	75%	44%	47%	52%	48%	98%

Table 6.6: Comparison with TailorNet retrained with my dataset with high frequency discrete anchors (*w/ anchor*), after applying its post-processing (*-post*), and without them (*w/o anchor*). My method reduces the error metrics by 44% to 98%.

6.5.6 Qualitative Results

I compare my model (*16k* in Table 6.4) qualitatively with previous works [11, 180, 181]. A comparison against Tailornet trained on my dataset is shown in Fig. 6.8. In a few examples where the garment roughly fits the body (Column 1, 2), TailorNet successfully predicts good results without too many intersections. However, my prediction results are still visually better due to realization of steeper wrinkles. In other cases, TailorNet fails to provide collision-free results, and the garments are unrealistically crumpled. In contrast, my method provides realistic folds and wrinkles of the garment that follow the dynamics with the body shape.

I also tried my best to compare my results with other two works where code is not available [180, 181]. I took their original figures and generated my drapes for similar body shape. Based on the results in Fig 6.9, I hypothesize that their

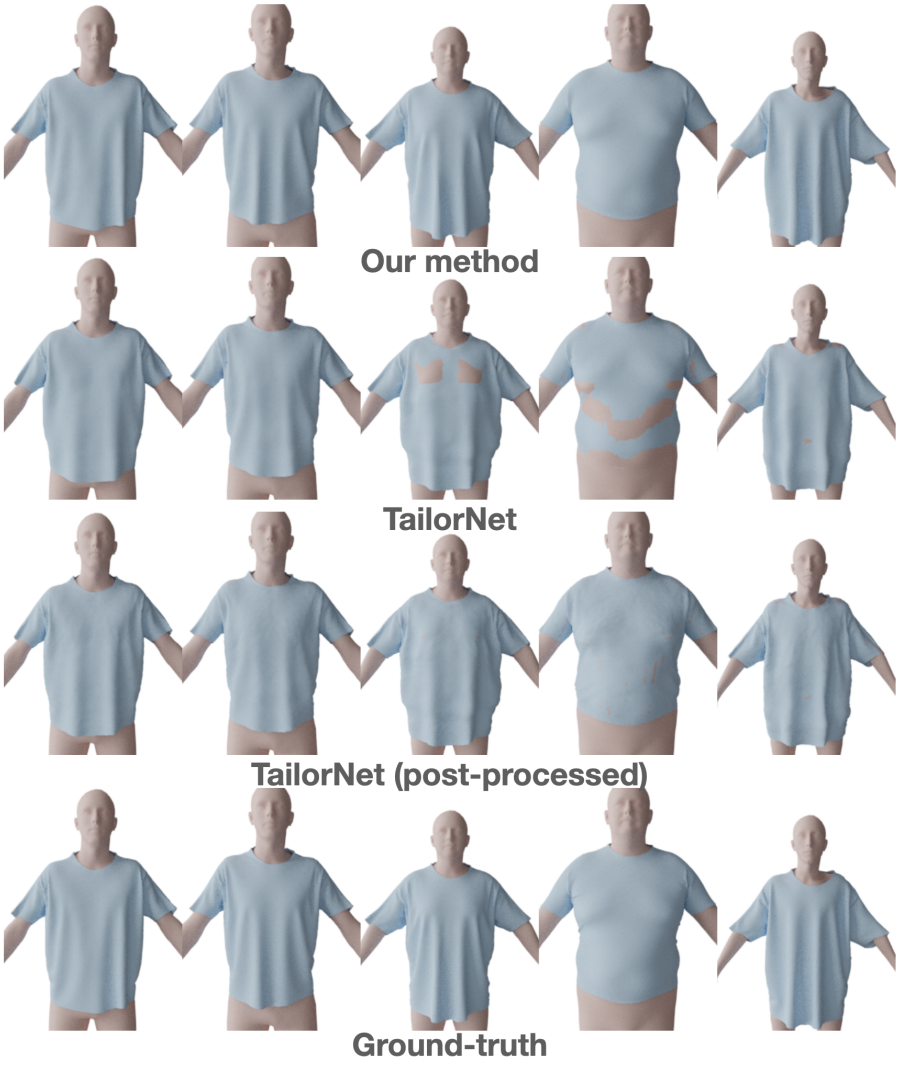


Figure 6.8: Qualitative comparison with TailorNet. My model captures better garment shapes regarding global fold structures, normal consistency, fine wrinkle details, and collision avoidance.

per-vertex loss generates smoother garments with less rich wrinkles. However, I should note that there are substantial differences in terms of T-shirt size, topology and materials, so this comparison is far from definitive.

6.5.7 Generalization to Different Garment Sizes

This section shows the potential of my model to generalize to different sizes of the garment. I simulate a smaller T-shirt of the same topology on the original

[181] Mine [180] Mine

Figure 6.9: Qualitative comparison with [180, 181]. My model generates finer wrinkles around the waist and armpits.

20,000 body samples. I train three different models based on 16,000 small T-shirts, 16,000 original T-shirts, and 32,000 T-shirt of both sizes, and tested them either against the small or original size set, both of them with 2,000 samples. When the network is trained on all the data, the T-shirt size is indicated to the network by simply introducing a flag to the encoded features without any further modifications. As shown in Table 6.7, the model trained on both sizes preserves similar accuracy to the ones trained separately, only slightly worse due to the higher complexity of the task. To improve the results, one possible solution is to train an auto-encoder for a set of garment templates and inject the latent code as the feature. I leave this exploration as future work.

Train-Test Set	ME (cm)	$\mathcal{L}_l$ (cm)	$\mathcal{L}_e$ (%)	$\mathcal{L}_s$	$\mathcal{L}_d$	p (%)
Small-Small	0.34	0.16	9.26	1.1e-3	4.5e-3	0.09
Both-Small	0.56	0.19	11.28	1.3e-3	5.5e-3	0.41
Original-Original	0.33	0.16	9.21	1.0e-3	4.5e-3	0.05
Both-Original	0.34	0.16	9.43	1.1e-3	4.6e-3	0.05

Table 6.7: Comparison for models trained on different sizes of the garments; A-B means train on A and tested on B. The model trained on both sizes achieves comparable results.

6.6 Conclusion

I propose a novel network structure and semi-supervised framework to learn realistic fit-accurate garment draping on a wide range of human body sizes. I utilize physics-enforced loss functions and a more expressive GCN decoder during the initial training. Next, a self-correcting optimization method that minimizes the potential energy is proposed to fix the remaining violation of physical laws. Retraining my original network with the optimized samples makes its predictions even more physically accurate. Experimental results show that my method outperforms the state of the art regarding physical realism, dynamics-dependent wrinkle formation, and collision avoidance.

One limitation of this work is that my model currently only supports a canonical body pose. In future work I am planning to support multiple poses at test time by simulating my data under multiple poses and using the unposing technique from [11].

Chapter 7: Conclusion

In this dissertation, I proposed various methods for reconstructing human body and garment from images or videos, as well as synthesizing new garments given the body mesh. These methods enables shape-aware body recovery from multi-view images, accurate material retrieval using optimization, faithful garment reconstruction together with body and material estimation from videos, efficient and scalable distributed simulation, and fast and realistic garment prediction on human bodies. I have shown their high effectiveness and accuracy in extensive experiments. The proposed work can greatly boost the efficiency and realism of simulation-based virtual try-on systems.

7.1 Summary of Results

To train a shape-aware body estimation model, Chapter 2 introduces a novel multi-view multi-stage framework. This framework is scalable and can take arbitrary number of views as input. Moreover, it describes a training data generation pipeline using physically-based simulation. It is critical for shape estimation and regularization of end-effectors that real-world datasets cannot provide. Experiments show that this method benefits from the simulated dataset generated from this pro-

posed pipeline and outperforms existing methods on real-world images, especially on shape estimations.

Material cloning from real world to virtual world is critical to attaining realism of virtual try-on systems. To enable a first-order estimation of garment materials, Chapter 3 presents a differentiable cloth simulator that can provide the gradients of the cloth simulation function. The gradient of the dynamic collision handling is explicitly derived. In order to obtain gradients of large formulated functions, implicit differentiation is used instead of backpropagating step by step. Experiments show that my backpropagation is two orders of magnitude faster than the baseline method. The proposed differentiable simulation can be used in a number of inverse problems, when the environmental setup is known.

To further combine the two correlated objectives above for an end-to-end method with higher estimation accuracy, Chapter 4 proposes an end-to-end learning model for estimating body and garment material from RGB videos. In order to maximize the multi-tasking benefits, human body and the garment shape are jointly learned as features for the material prediction. To reduce the degree of freedom of cloth due to its highly dynamic and deformable nature of cloth, a two-level auto-encoder to represent garments has been proposed in a hierarchical structure. Using this network, it also becomes possible to smoothly transition the geometry between different garment topologies. During the estimation, a feedback loop is introduced to correctly refine the body estimation using the garment prediction. Experiments show that this proposed system has the highest estimation accuracy and can be easily generalized to unseen input.

Cloth simulations, widely used in computer animation and apparel design, can be computationally expensive for real-time applications. Some parallelization techniques have been proposed for visual simulation of cloth using CPU or GPU clusters and often rely on parallelization using spatial domain decomposition techniques that have a large communication overhead. To avoid a large overhead, Chapter 5 introduces a novel temporal-domain parallelization method for performance-demanding tasks. I parallelize the cloth simulation in temporal domain by using a faster simulation result on coarser meshes as an initialization, resulting in accelerated computation within each temporal computational block and minimal communication overhead. An iterative detail recovery algorithm is designed to minimize the visual artifacts due to the inconsistency between two resolutions. This proposed method has nearly linear scaling on manycore clusters and achieves more runtime performance advantages, when compared to previous parallel methods with a higher number of computing cores.

Finally, to train an efficient and accurate prediction model of garments to provide the final try-on results, Chapter 6 proposes a novel semi-supervised model to learn physically-correct garment draping on a large variation of human body shapes. Several loss functions are incorporated to increase the physical awareness of the network, and a new GCN decoder is used for higher expressiveness. Moreover, a self-correcting optimization method is adopted to minimize the potential energy of the prediction and remove the remaining collision with the human body. Retraining this proposed network with the optimized samples can yield better predictions. Experiments show that this novel method can provide better draping predictions

than previous works in terms of simulated wrinkles and folds, it can also cover a large distribution of body shapes, while achieving higher speed and accuracy compared to physics-based cloth simulation.

7.2 Limitations

There are four main limitations in my proposed methods. First, there is a sim-to-real gap that can hinder the virtual world to reproduce exactly the same visual effect as those in the real world. Common sources of this gap include lighting, human skin texture, mesh resolution, and spatial relation with the background and other objects in the scene. Simulation settings, such as the discretization scheme, the physical governing laws, and the material model, can also affect the final results. When the simulated data is used for training network models, this gap may potentially affect its generalization to unseen real-world images or videos. While higher-quality training data with more complex settings can certainly reduces the gap, the labor cost of doing so increases as well. Although the current results show that models can already perform well when trained with the simulation data, the potential improvements given by more realistic training data is not yet quantified.

Second, the high computational cost of cloth simulation is also a challenge. When simulating garments for learning purposes or optimizations, I want the simulation to be as fast as possible so that I can obtain the maximum amount of training data or the maximum number of iterations within a given amount of time. When the simulation is used for visual demonstration, real-time response from the user

interaction is critical. Although the proposed method in Chapter 5 achieves near real-time performance, the single-frame simulation runtime is not reduced in this framework, leading to a high latency feedback.

Next, a parametric generation model for universal garments is still missing. An ideal garment generation model should be able to account for different variations, including sizes and lengths, topology, fitness to any given body shape, sewing patterns, or even accessory locations and geometry. Chapter 4 introduces the first generation model that unites garments of different topology and sizes using point clouds. However, this model cannot fully represent garments that have multiple layers or self-folding, because it does not store connectivity information between points. Moreover, it does not currently support multiple separated garments and sewing seams for the same reason. However, such geometric information is critical, as it directly affects the physical behavior of the reconstructed and learned garment models.

At last, there remains a large visual disparity and physical difference between simulation results and network prediction results when it comes to garment synthesis. Although the proposed method in Chapter 6 successfully connects simulation realism to the network prediction by introducing dynamical constraints in the training for more realistic predictions, the coverage of the current model is still limited, when compared to all possible combinations of human poses and shapes, garment sizes, topology and materials. Given a specific type of garment, the best model I can get in theory will be able to present different wrinkle formations according to different input of the human body and the garment material. However, it requires

an unrealistic amount of captured or simulated data used for training to cover the entire spectrum spanned by the control variables. As it is practically impossible to include every sample in the training datasets, an effective and general decoupling algorithm between different factors is necessary. But, this problem is currently under-explored.

7.3 Future Work

While the proposed methods have been proven to be effective in the reconstruction and the synthesis tasks regarding garments dressed on human bodies, there are still several potential improvements and challenges that need to be addressed. Below I suggest a few possible future directions regarding each proposed limitation described in Section 7.2.

Training data. More research can be conducted to minimize or mitigate the sim-to-real gap in training data distributions. Chapter 2 and 4 use simulated data for training the models due to ease of collecting ground-truth labels. While cloth simulation provides a large diversity of training samples with ground-truth parameters, a more systematic garment design and registration method are needed to minimize the visual perception difference between real-world images and synthetic ones. In addition, other variables such as hair, skin color, and 3D backgrounds can also influence the perceived realism of the synthetic data, but implementing them requires a more complex data generation pipeline. With the recent progress in image style transfer and translation using GAN [81], a promising direction is to transfer the

appearance of realistic images to the rendered simulation models to further improve the learning results.

Learning algorithms and architectures. A lot of options are adopted regarding network architectures being used in different tasks. Chapter 2 introduces a shared-weight prediction correction framework to integrate multi-view information, while Chapter 4 jointly learns the body and the garment shape together with a feed-back loop and feeds the single-frame features to an LSTM for material estimation. One key problem remains to be solved is to what extent do different network and framework choices affect the final result, and whether or not there is a specialized architecture that is generally beneficial to the capturing task regarding garment properties.

Parametric garment models. More improvements can be made regarding the parametric garment modeling introduced in Chapter 4. First, encoding multi-layer information can be achieved by adding structural prior to the garment model so that more complex structures of garments can be supported. For example, one can encode a tree structure to represent different garment parts from bodice to accessories. Second, adding curvature information such as normal in addition to 3D location can enable support of multi-fold features. A normal vector is extremely helpful to differentiate two pieces of cloth stacked together with different orientations. Taking a step further, encoding the texture space coordinates together can even provide the connectivity information between vertices, potentially allowing easier reconstruction from point clouds to meshes and smoother results.

Human body representation. Currently, the parametric human body model in Chapter 2, 4, and 6 uses only linear blend shapes and principal components to represent thehuman body. An intrinsic drawback of this approach is that it cannot model deformation of the body when encountering geometric constraints such as collisions. It would be of great value to investigate more complex and realistic body models as a basic building block of all estimation and prediction modules.

Visual rendering and synthesis. To further speed up the cloth simulation, I can either introduce vectorization or implement multiple parallelization schemes at the same time. The current simulation implementation in Chapter 3 is not optimized for large-scale vectorized operations, leading to imperfect runtime performance. This issue can be improved by a specialized, optimized simulation system based solely on large tensor operations. Chapter 5 also proposes a method to accelerate the computation, which is the time-domain parallelization. This method can naturally be combined with GPU-based spatial-domain parallelization to reduce the per-frame computation time, which further decreases the latency and increases the scalability in terms of the number of computational units.

Generalization and robustness. Disentanglement of multi-dimensional input factors is the key to boost the generalization ability for the method proposed in Chapter 6. Regarding data representation, principal component analysis (PCA) or tangent space displacement representation can be potential directions to disentangle body pose and shape variations. Another promising future improvement can be training refinement networks conditioned on body pose parameters or the corre-

sponding global transformations. While single-frame draping prediction is far from being solved, extending the current work to learning continuous garment motion is the ultimate goal towards real-time animated virtual try-on. This will require not only the encoding of the geometric constraints, but also the capturing of different temporal patterns dependent on the fabric material type, and can eventually become a differentiable simulation in a learning-based way.

Bibliography

[1] Stevie Giovanni, Yeun Chul Choi, Jay Huang, Eng Tat Khoo, and KangKang Yin. Virtual try-on using kinect and hd camera. In *International Conference on Motion in Games*, pages 55–65. Springer, 2012.

[2] Thiemo Alldieck, Marcus Magnor, Weipeng Xu, Christian Theobalt, and Gerard Pons-Moll. Video based reconstruction of 3d people models. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8387–8397, 2018.

[3] Angjoo Kanazawa, Michael J Black, David W Jacobs, and Jitendra Malik. End-to-end recovery of human shape and pose. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7122–7131, 2018.

[4] Gul Varol, Duygu Ceylan, Bryan Russell, Jimei Yang, Ersin Yumer, Ivan Laptev, and Cordelia Schmid. Bodynet: Volumetric inference of 3d human body shapes. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 20–36, 2018.

[5] Zerong Zheng, Tao Yu, Yixuan Wei, Qionghai Dai, and Yebin Liu. Deephuman: 3d human reconstruction from a single image. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 7739–7749, 2019.

[6] Shunsuke Saito, Zeng Huang, Ryota Natsume, Shigeo Morishima, Angjoo Kanazawa, and Hao Li. Pifu: Pixel-aligned implicit function for high-resolution clothed human digitization. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2304–2314, 2019.

[7] Albert Pumarola, Jordi Sanchez-Riera, Gary Choi, Alberto Sanfeliu, and Francesc Moreno-Noguer. 3dpeople: Modeling the geometry of dressed humans. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2242–2251, 2019.

[8] Erhan Gundogdu, Victor Constantin, Amrollah Seifoddini, Minh Dang, Mathieu Salzmann, and Pascal Fua. Garnet: A two-stream network for fast and

accurate 3d cloth draping. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 8739–8748, 2019.

[9] Tuanfeng Y Wang, Duygu Ceylan, Jovan Popovic, and Niloy J Mitra. Learning a shared shape space for multimodal garment design. *arXiv preprint arXiv:1806.11335*, 2018.

[10] Bharat Lal Bhatnagar, Garvita Tiwari, Christian Theobalt, and Gerard Pons-Moll. Multi-garment net: Learning to dress 3d people from images. In *2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019*, pages 5419–5429. IEEE, 2019.

[11] Chaitanya Patel, Zhouyingcheng Liao, and Gerard Pons-Moll. Tailornet: Predicting clothing in 3d as a function of human pose, shape and garment style. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, jun 2020.

[12] R Daněřek, Endri Dibra, Cengiz Öztireli, Remo Ziegler, and Markus Gross. Deepgarment: 3d garment shape estimation from a single image. In *Computer Graphics Forum*, volume 36, pages 269–280. Wiley Online Library, 2017.

[13] Filipe de Avila Belbute-Peres, Kevin A. Smith, Kelsey Allen, Josh Tenenbaum, and J. Zico Kolter. End-to-end differentiable physics for learning and control. In *Advances in Neural Information Processing Systems*, 2018.

[14] Jonas Degrave, Michiel Hermans, Joni Dambre, and Francis wyffels. A differentiable physics engine for deep learning in robotics. *Frontiers in Neurorobotics*, 13, 2019.

[15] Marc Toussaint, Kelsey Allen, Kevin Smith, and Joshua Tenenbaum. Differentiable physics and stable modes for tool-use and manipulation planning. In *Robotics: Science and Systems (RSS)*, 2018.

[16] Yuanming Hu, Jiancheng Liu, Andrew Spielberg, Joshua B. Tenenbaum, William T. Freeman, Jiajun Wu, Daniela Rus, and Wojciech Matusik. ChainQueen: A real-time differentiable physical simulator for soft robotics. In *International Conference on Robotics and Automation (ICRA)*, 2019.

[17] Connor Schenck and Dieter Fox. SPNets: Differentiable fluid dynamics for deep neural networks. In *Conference on Robot Learning (CoRL)*, 2018.

[18] Marco Cusumano-Towner, Arjun Singh, Stephen Miller, James F. O'Brien, and Pieter Abbeel. Bringing clothing into desired configurations with limited perception. In *International Conference on Robotics and Automation (ICRA)*, 2011.

[19] Stephen Miller, Jur van den Berg, Mario Fritz, Trevor Darrell, Kenneth Y. Goldberg, and Pieter Abbeel. A geometric approach to robotic laundry folding. *I. J. Robotics Res.*, 31(2), 2012.

[20] Katherine L. Bouman, Bei Xiao, Peter Battaglia, and William T. Freeman. Estimating the material properties of fabric from video. In *International Conference on Computer Vision (ICCV)*, 2013.

[21] Shan Yang, Junbang Liang, and Ming C Lin. Learning-based cloth material recovery from video. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4383–4393, 2017.

[22] Gerard Pons-Moll, Sergi Pujades, Sonny Hu, and Michael J Black. Clothcap: Seamless 4d clothing capture and retargeting. *ACM Transactions on Graphics (TOG)*, 36(4):1–15, 2017.

[23] Zorah Lähner, Daniel Cremers, and Tony Tung. DeepWrinkles: Accurate and realistic clothing modeling. In *European Conference on Computer Vision (ECCV)*, 2018.

[24] Rony Goldenthal, David Harmon, Raanan Fattal, Michel Bercovier, and Eitan Grinspun. Efficient simulation of inextensible cloth. In *ACM SIGGRAPH 2007 papers*, pages 49–es. 2007.

[25] Dongliang Zhang and Matthew MF Yuen. Cloth simulation using multilevel meshes. *Computers & Graphics*, 25(3):383–389, 2001.

[26] Robert Bridson, Ronald Fedkiw, and John Anderson. Robust treatment of collisions, contact and friction for cloth animation. *ACM Transactions on Graphics (ToG)*, 21(3):594–603, 2002.

[27] David Baraff and Andrew Witkin. Large steps in cloth simulation. In *SIGGRAPH*, 1998.

[28] Assaf Neuberger, Eran Borenstein, Bar Hilleli, Eduard Oks, and Sharon Alpert. Image based virtual try-on network from unpaired data. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5184–5193, 2020.

[29] Xintong Han, Zuxuan Wu, Zhe Wu, Ruichi Yu, and Larry S Davis. Viton: An image-based virtual try-on network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7543–7552, 2018.

[30] Matthew Loper, Naureen Mahmood, Javier Romero, Gerard Pons-Moll, and Michael J Black. Smpl: A skinned multi-person linear model. *ACM transactions on graphics (TOG)*, 34(6):1–16, 2015.

[31] Dushyant Mehta, Srinath Sridhar, Oleksandr Sotnychenko, Helge Rhodin, Mohammad Shafiei, Hans-Peter Seidel, Weipeng Xu, Dan Casas, and Christian Theobalt. Vnect: Real-time 3d human pose estimation with a single rgb camera. *ACM Transactions on Graphics (TOG)*, 36(4):44, 2017.

[32] Angjoo Kanazawa, Michael J. Black, David W. Jacobs, and Jitendra Malik. End-to-end recovery of human shape and pose. In *Computer Vision and Pattern Regognition (CVPR)*, 2018.

[33] Georgios Pavlakos, Xiaowei Zhou, Konstantinos G Derpanis, and Kostas Daniilidis. Coarse-to-fine volumetric prediction for single-image 3d human pose. In *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*, pages 1263–1272. IEEE, 2017.

[34] Bugra Tekin, Pablo Marquez Neila, Mathieu Salzmann, and Pascal Fua. Learning to fuse 2d and 3d image cues for monocular body pose estimation. In *International Conference on Computer Vision (ICCV)*, number EPFL-CONF-230311, 2017.

[35] Denis Tome, Christopher Russell, and Lourdes Agapito. Lifting from the deep: Convolutional 3d pose estimation from a single image. *CVPR 2017 Proceedings*, pages 2500–2509, 2017.

[36] Xingyi Zhou, Qixing Huang, Xiao Sun, Xiangyang Xue, and Yichen Wei. Weaklysupervised transfer for 3d human pose estimation in the wild. In *IEEE International Conference on Computer Vision*, volume 206, page 3, 2017.

[37] Xingyi Zhou, Xiao Sun, Wei Zhang, Shuang Liang, and Yichen Wei. Deep kinematic pose regression. In *European Conference on Computer Vision*, pages 186–201. Springer, 2016.

[38] Alexandru O Balan, Leonid Sigal, Michael J Black, James E Davis, and Horst W Haussecker. Detailed human shape and pose from images. In *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*, pages 1–8. IEEE, 2007.

[39] Yu Chen, Tae-Kyun Kim, and Roberto Cipolla. Inferring 3d shapes and deformations from single views. In *European Conference on Computer Vision*, pages 300–313. Springer, 2010.

[40] Endri Dibra, Himanshu Jain, Cengiz Öztireli, Remo Ziegler, and Markus Gross. Hs-nets: Estimating human body shape from silhouettes with convolutional neural networks. In *3D Vision (3DV), 2016 Fourth International Conference on*, pages 108–117. IEEE, 2016.

[41] Peng Guan, Alexander Weiss, Alexandru O Balan, and Michael J Black. Estimating human shape and pose from a single image. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 1381–1388. IEEE, 2009.

[42] Nils Hasler, Hanno Ackermann, Bodo Rosenhahn, Thorsten Thormählen, and Hans-Peter Seidel. Multilinear pose and body shape estimation of dressed subjects from image sets. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 1823–1830. IEEE, 2010.

[43] Arjun Jain, Thorsten Thormählen, Hans-Peter Seidel, and Christian Theobalt. Moviereshape: Tracking and reshaping of humans in videos. In *ACM Transactions on Graphics (TOG)*, volume 29, page 148. ACM, 2010.

[44] J Tan, Ignas Budvytis, and Roberto Cipolla. Indirect deep structured learning for 3d human body shape and pose prediction. In *BMVC*, volume 3, page 6, 2017.

[45] Hsiao-Yu Tung, Hsiao-Wei Tung, Ersin Yumer, and Katerina Fragkiadaki. Self-supervised learning of motion capture. In *Advances in Neural Information Processing Systems*, pages 5236–5246, 2017.

[46] Nils Hasler, Carsten Stoll, Bodo Rosenhahn, Thorsten Thormählen, and Hans-Peter Seidel. Estimating body shape of dressed humans. *Computers & Graphics*, 33(3):211–216, 2009.

[47] Stefanie Wuhrer, Leonid Pishchulin, Alan Brunton, Chang Shu, and Jochen Lang. Estimation of human body shape and posture under clothing. *Computer Vision and Image Understanding*, 127:31–42, 2014.

[48] Jinlong Yang, Jean-Sébastien Franco, Franck Hétroy-Wheeler, and Stefanie Wuhrer. Estimation of human body shape in motion with wide clothing. In *European Conference on Computer Vision*, pages 439–454. Springer, 2016.

[49] Chao Zhang, Sergi Pujades, Michael Black, and Gerard Pons-Moll. Detailed, accurate, human shape estimation from clothed 3d scan sequences. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, 2017.

[50] Alexandru O Bălan and Michael J Black. The naked truth: Estimating body shape under clothing. In *European Conference on Computer Vision*, pages 15–29. Springer, 2008.

[51] Federica Bogo, Angjoo Kanazawa, Christoph Lassner, Peter Gehler, Javier Romero, and Michael J Black. Keep it smpl: Automatic estimation of 3d human pose and shape from a single image. In *European Conference on Computer Vision*, pages 561–578. Springer, 2016.

[52] Christoph Lassner, Javier Romero, Martin Kiefel, Federica Bogo, Michael J Black, and Peter V Gehler. Unite the people: Closing the loop between 3d and 2d human representations. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, volume 2, page 3, 2017.

[53] Matthew Trumble, Andrew Gilbert, Charles Malleson, Adrian Hilton, and John Collomosse. Total capture: 3d human pose estimation fusing video and inertial sensors. In *Proceedings of 28th British Machine Vision Conference*, pages 1–13, 2017.

[54] Juan Carlos Núñez, Raúl Cabido, José F Vélez, Antonio S Montemayor, and Juan José Pantrigo. Multiview 3d human pose estimation using improved least-squares and lstm networks. *Neurocomputing*, 323:335–343, 2019.

[55] Helge Rhodin, Mathieu Salzmann, and Pascal Fua. Unsupervised geometry-aware representation for 3d human pose estimation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 750–767, 2018.

[56] Matthew Trumble, Andrew Gilbert, Adrian Hilton, and John Collomosse. Deep autoencoder for combined human pose estimation and body model upscaling. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 784–800, 2018.

[57] Georgios Pavlakos, Xiaowei Zhou, Konstantinos G Derpanis, and Kostas Daniilidis. Harvesting multiple views for marker-less 3d human pose annotations. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6988–6997, 2017.

[58] Denis Tome, Matteo Toso, Lourdes Agapito, and Chris Russell. Rethinking pose in 3d: Multi-stage refinement and recovery for markerless motion capture. In *2018 International Conference on 3D Vision (3DV)*, pages 474–483. IEEE, 2018.

[59] Christopher B Choy, Danfei Xu, JunYoung Gwak, Kevin Chen, and Silvio Savarese. 3d-r2n2: A unified approach for single and multi-view 3d object reconstruction. In *European conference on computer vision*, pages 628–644. Springer, 2016.

[60] Mohamed Omran, Christoph Lassner, Gerard Pons-Moll, Peter Gehler, and Bernt Schiele. Neural body fitting: Unifying deep learning and model based human pose and shape estimation. In *2018 International Conference on 3D Vision (3DV)*, pages 484–494. IEEE, 2018.

[61] Georgios Pavlakos, Luyang Zhu, Xiaowei Zhou, and Kostas Daniilidis. Learning to estimate 3d human pose and shape from a single color image. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 459–468, 2018.

[62] Nikos Kolotouros, Georgios Pavlakos, and Kostas Daniilidis. Convolutional mesh regression for single-image human shape reconstruction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4501–4510, 2019.

[63] Wenzheng Chen, Huan Wang, Yangyan Li, Hao Su, Zhenhua Wang, Changhe Tu, Dani Lischinski, Daniel Cohen-Or, and Baoquan Chen. Synthesizing training images for boosting human 3d pose estimation. In *3D Vision (3DV), 2016 Fourth International Conference on*, pages 479–488. IEEE, 2016.

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

[65] CMU. Carnegie-mellon mocap database. created with funding from nsf eia-0196217, 2003.

[66] Kathleen M Robinette, Sherri Blackwell, Hein Daanen, Mark Boehmer, and Scott Fleming. Civilian american and european surface anthropometry resource (caesar), final report. volume 1. summary. Technical report, SYTRONICS INC DAYTON OH, 2002.

[67] Hosnieh Sattar, Gerard Pons-Moll, and Mario Fritz. Fashion is taking shape: Understanding clothing preference based on body shape from online sources. In *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 968–977. IEEE, 2019.

[68] Thiemo Alldieck, Marcus Magnor, Bharat Lal Bhatnagar, Christian Theobalt, and Gerard Pons-Moll. Learning to reconstruct people in clothing from a single rgb camera. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1175–1186, 2019.

[69] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[70] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

[71] Dushyant Mehta, Helge Rhodin, Dan Casas, Pascal Fua, Oleksandr Sotnychenko, Weipeng Xu, and Christian Theobalt. Monocular 3d human pose estimation in the wild using improved cnn supervision. In *3D Vision (3DV), 2017 Fifth International Conference on*. IEEE, 2017.

[72] Catalin Ionescu, Dragos Papava, Vlad Olaru, and Cristian Sminchisescu. Human3. 6m: Large scale datasets and predictive methods for 3d human sensing in natural environments. *IEEE transactions on pattern analysis and machine intelligence*, 36(7):1325–1339, 2014.

[73] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *European conference on computer vision*, pages 630–645. Springer, 2016.

[74] Rahul Narain, Armin Samii, and James F. O'Brien. Adaptive anisotropic remeshing for cloth simulation. *ACM Trans. Graph.*, 31(6), 2012.

[75] Dushyant Mehta, Helge Rhodin, Dan Casas, Pascal Fua, Oleksandr Sotnychenko, Weipeng Xu, and Christian Theobalt. Monocular 3d human pose estimation in the wild using improved cnn supervision. In *3D Vision (3DV), 2017 International Conference on*, pages 506–516. IEEE, 2017.

[76] Helge Rhodin, Nadia Robertini, Dan Casas, Christian Richardt, Hans-Peter Seidel, and Christian Theobalt. General automatic human shape and motion capture using volumetric contour cues. In *European conference on computer vision*, pages 509–526. Springer, 2016.

[77] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross B. Girshick. Mask R-CNN. *CoRR*, abs/1703.06870, 2017.

[78] Gregory Rogez, Philippe Weinzaepfel, and Cordelia Schmid. Lcr-net: Localization-classification-regression for human pose. In *CVPR 2017-IEEE Conference on Computer Vision & Pattern Recognition*, 2017.

[79] Xiao Sun, Jiaxiang Shang, Shuang Liang, and Yichen Wei. Compositional human pose regression. In *The IEEE International Conference on Computer Vision (ICCV)*, volume 2, page 7, 2017.

[80] Rishabh Dabral, Anurag Mundhada, Uday Kusupati, Safeer Afaque, Abhishek Sharma, and Arjun Jain. Learning 3d human pose from structure and motion. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 668–683, 2018.

[81] Franziska Mueller, Florian Bernard, Oleksandr Sotnychenko, Dushyant Mehta, Srinath Sridhar, Dan Casas, and Christian Theobalt. Ganerated hands for real-time 3d hand tracking from monocular rgb. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 49–59, 2018.

[82] Richard W Cottle. *Linear Complementarity Problem*. Springer, 2009.

[83] Michael Bradley Cline. *Rigid Body Simulation with Contact and Constraints*. PhD thesis, University of British Columbia, 2002.

[84] Damian Mrowca, Chengxu Zhuang, Elias Wang, Nick Haber, Fei-Fei Li, Josh Tenenbaum, and Daniel L. Yamins. Flexible neural representation for physics prediction. In *Advances in Neural Information Processing Systems*, 2018.

[85] Yunzhu Li, Jiajun Wu, Russ Tedrake, Joshua B. Tenenbaum, and Antonio Torralba. Learning particle dynamics for manipulating rigid bodies, deformable objects, and fluids. In *International Conference on Learning Representations (ICLR)*, 2019.

[86] John Ingraham, Adam Riesselman, Chris Sander, and Debora Marks. Learning protein structure with a differentiable simulator. In *International Conference on Learning Representations (ICLR)*, 2019.

[87] Jonathan Tompson, Kristofer Schlachter, Pablo Sprechmann, and Ken Perlin. Accelerating Eulerian fluid simulation with convolutional networks. In *International Conference on Machine Learning (ICML)*, 2017.

[88] J Nathan Kutz. Deep learning in fluid dynamics. *Journal of Fluid Mechanics*, 814, 2017.

[89] Jeremy Morton, Antony Jameson, Mykel J Kochenderfer, and Freddie Witherden. Deep dynamical modeling and control of unsteady fluid flows. In *Advances in Neural Information Processing Systems*, 2018.

[90] Igor Santesteban, Miguel A. Otaduy, and Dan Casas. Learning-based animation of clothing for virtual try-on. In *Eurographics*, 2019.

[91] Olaf Etzmuß, Michael Keckeisen, and Wolfgang Straßer. A fast finite element solution for cloth modelling. In *Pacific Conference on Computer Graphics and Applications*, 2003.

[92] David Harmon, Etienne Vouga, Rasmus Tamstorf, and Eitan Grinspun. Robust treatment of simultaneous collisions. *ACM Trans. Graph.*, 27(3), 2008.

[93] Huamin Wang, James F. O'Brien, and Ravi Ramamoorthi. Data-driven elastic models for cloth: Modeling and measurement. *ACM Trans. Graph.*, 30(4), 2011.

[94] Eitan Grinspun, Anil N Hirani, Mathieu Desbrun, and Peter Schröder. Discrete shells. In *Symposium on Computer Animation*, 2003.

[95] Benoit Steiner, Zachary DeVito, Soumith Chintala, Sam Gross, Adam Paszke, Francisco Massa, Adam Lerer, Gregory Chanan, Zeming Lin, Edward Yang, et al. PyTorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, 2019.

[96] Min Tang, Dinesh Manocha, and Ruofeng Tong. Fast continuous collision detection using deforming non-penetration filters. In *Symposium on Interactive 3D Graphics and Games*, 2010.

[97] Brandon Amos and J. Zico Kolter. OptNet: Differentiable optimization as a layer in neural networks. In *International Conference on Machine Learning (ICML)*, 2017.

[98] Eric Liang, Richard Liaw, Robert Nishihara, Philipp Moritz, Roy Fox, Ken Goldberg, Joseph E. Gonzalez, Michael I. Jordan, and Ion Stoica. RLlib: Abstractions for distributed reinforcement learning. In *International Conference on Machine Learning (ICML)*, 2018.

[99] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 652–660, 2017.

[100] Romain Casati, Gilles Daviet, and Florence Bertails-Descoubes. *Inverse elastic cloth design with contact and friction.* PhD thesis, Inria Grenoble Rhône-Alpes, Université de Grenoble, 2016.

[101] Bernd Bickel, Moritz Bächer, Miguel A Otaduy, Hyunho Richard Lee, Hanspeter Pfister, Markus Gross, and Wojciech Matusik. Design and fabrication of materials with desired deformation behavior. *ACM Transactions on Graphics (TOG)*, 29(4):1–10, 2010.

[102] Raquel Vidaurre, Dan Casas, Elena Garces, and Jorge Lopez-Moreno. Brdf estimation of complex materials with nested learning. In *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1347–1356. IEEE, 2019.

[103] Giuseppe Claudio Guarnera, Peter Hall, Alain Chesnais, and Mashhuda Glencross. Woven fabric model creation from a single image. *ACM Transactions on Graphics (TOG)*, 36(5):1–13, 2017.

[104] Wenyan Bi, Peiran Jin, Hendrikje Nienborg, and Bei Xiao. Estimating mechanical properties of cloth from videos using dense motion trajectories: Human psychophysics and machine learning. *Journal of vision*, 18(5):12–12, 2018.

[105] Wenyan Bi and Bei Xiao. Perceptual constancy of mechanical properties of cloth under variation of external forces. In *Proceedings of the ACM symposium on applied perception*, pages 19–23, 2016.

[106] Abdullah Haroon Rasheed, Victor Romero, Florence Bertails-Descoubes, Stefanie Wuhrer, Jean-Sébastien Franco, and Arnaud Lazarus. Learning to measure the static friction coefficient in cloth contact. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9912–9921, 2020.

[107] Eder Miguel, Rasmus Tamstorf, Derek Bradley, Sara C Schvartzman, Bernhard Thomaszewski, Bernd Bickel, Wojciech Matusik, Steve Marschner, and Miguel A Otaduy. Modeling and estimation of internal friction in cloth. *ACM Transactions on Graphics (TOG)*, 32(6):1–10, 2013.

[108] Katherine L Bouman, Bei Xiao, Peter Battaglia, and William T Freeman. Estimating the material properties of fabric from video. In *Proceedings of the IEEE international conference on computer vision*, pages 1984–1991, 2013.

[109] David Clyde, Joseph Teran, and Rasmus Tamstorf. Modeling and data-driven parameter estimation for woven fabrics. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 1–11, 2017.

[110] Shan Yang, Tanya Ambert, Zherong Pan, Ke Wang, Licheng Yu, Tamara Berg, and Ming C Lin. Detailed garment recovery from a single-view image. *arXiv preprint arXiv:1608.01250*, 2016.

[111] Eder Miguel, Derek Bradley, Bernhard Thomaszewski, Bernd Bickel, Wojciech Matusik, Miguel A Otaduy, and Steve Marschner. Data-driven estimation of cloth simulation models. In *Computer Graphics Forum*, volume 31, pages 519–528. Wiley Online Library, 2012.

[112] Kiran S Bhat, Christopher D Twigg, Jessica K Hodgins, Pradeep Khosla, Zoran Popovic, and Steven M Seitz. Estimating cloth simulation parameters from video. 2003.

[113] Bin Zhou, Xiaowu Chen, Qiang Fu, Kan Guo, and Ping Tan. Garment modeling from a single image. In *Computer graphics forum*, volume 32, pages 85–91. Wiley Online Library, 2013.

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

[115] Derek Bradley, Tiberiu Popa, Alla Sheffer, Wolfgang Heidrich, and Tamy Boubekeur. Markerless garment capture. In *ACM SIGGRAPH 2008 papers*, pages 1–9. 2008.

[116] Thiemo Alldieck, Gerard Pons-Moll, Christian Theobalt, and Marcus Magnor. Tex2shape: Detailed full human body geometry from a single image. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2293–2303, 2019.

[117] Xiaowu Chen, Bin Zhou, Fei-Xiang Lu, Lin Wang, Lang Bi, and Ping Tan. Garment modeling with a depth camera. *ACM Trans. Graph.*, 34(6):203–1, 2015.

[118] Marc Habermann, Weipeng Xu, , Michael Zollhoefer, Gerard Pons-Moll, and Christian Theobalt. Deepcap: Monocular human performance capture using weak supervision. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, jun 2020.

[119] Zeng Huang, Yuanlu Xu, Christoph Lassner, Hao Li, and Tony Tung. Arch: Animatable reconstruction of clothed humans. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3093–3102, 2020.

[120] Heming Zhu, Yu Cao, Hang Jin, Weikai Chen, Dong Du, Zhangye Wang, Shuguang Cui, and Xiaoguang Han. Deep fashion3d: A dataset and benchmark for 3d garment reconstruction from single images. *arXiv preprint arXiv:2003.12753*, 2020.

[121] Boyi Jiang, Juyong Zhang, Yang Hong, Jinhao Luo, Ligang Liu, and Hujun Bao. Bcnet: Learning body and cloth shape from a single image. *arXiv preprint arXiv:2004.00214*, 2020.

[122] Tao Yu, Zerong Zheng, Yuan Zhong, Jianhui Zhao, Qionghai Dai, Gerard Pons-Moll, and Yebin Liu. Simulcap: Single-view human performance capture with cloth simulation. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5499–5509. IEEE, 2019.

[123] Garvita Tiwari, Bharat Lal Bhatnagar, Tony Tung, and Gerard Pons-Moll. Sizer: A dataset and model for parsing 3d clothing and learning size sensitive 3d clothing. *arXiv preprint arXiv:2007.11610*, 2020.

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

[125] Roman Klokov and Victor Lempitsky. Escape from cells: Deep kd-networks for the recognition of 3d point cloud models. In *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[126] Jiaxin Li, Ben M. Chen, and Gim Hee Lee. So-net: Self-organizing network for point cloud analysis. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[127] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *Advances in neural information processing systems*, pages 5099–5108, 2017.

[128] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E Sarma, Michael M Bronstein, and Justin M Solomon. Dynamic graph cnn for learning on point clouds. *ACM Transactions on Graphics (TOG)*, 38(5):1–12, 2019.

[129] Charles R. Qi, Wei Liu, Chenxia Wu, Hao Su, and Leonidas J. Guibas. Frustum pointnets for 3d object detection from rgb-d data. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[130] Yin Zhou and Oncel Tuzel. Voxelnet: End-to-end learning for point cloud based 3d object detection. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[131] Yangyan Li, Rui Bu, Mingchao Sun, Wei Wu, Xinhan Di, and Baoquan Chen. Pointcnn: Convolution on x-transformed points. In *Advances in neural information processing systems*, pages 820–830, 2018.

[132] Shih-En Wei, Varun Ramakrishna, Takeo Kanade, and Yaser Sheikh. Convolutional pose machines. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 4724–4732, 2016.

[133] Zhe Cao, Tomas Simon, Shih-En Wei, and Yaser Sheikh. Realtime multi-person 2d pose estimation using part affinity fields. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7291–7299, 2017.

[134] Yuanlu Xu, Song-Chun Zhu, and Tony Tung. Denserac: Joint 3d pose and shape estimation by dense render-and-compare. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 7760–7770, 2019.

[135] David Smith, Matthew Loper, Xiaochen Hu, Paris Mavroidis, and Javier Romero. Facsimile: Fast and accurate scans from an image in less than a second. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 5330–5339, 2019.

[136] Junbang Liang and Ming C Lin. Shape-aware human pose and shape reconstruction using multi-view images. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4352–4362, 2019.

[137] Nikos Kolotouros, Georgios Pavlakos, Michael J Black, and Kostas Daniilidis. Learning to reconstruct 3d human pose and shape via model-fitting in the loop. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2252–2261, 2019.

[138] Wenjuan Gong, Xuena Zhang, Jordi Gonzàlez, Andrews Sobral, Thierry Bouwmans, Changhe Tu, and El-hadi Zahzah. Human pose estimation from monocular images: A comprehensive survey. *Sensors*, 16(12):1966, 2016.

[139] Andrea Saltelli. Sensitivity analysis for importance assessment. *Risk analysis*, 22(3):579–590, 2002.

[140] Qingyang Tan, Zherong Pan, Lin Gao, and Dinesh Manocha. Realtime simulation of thin-shell deformable materials using cnn-based mesh embedding. *IEEE Robotics and Automation Letters*, 5(2):2325–2332, 2020.

[141] Theo Deprelle, Thibault Groueix, Matthew Fisher, Vladimir Kim, Bryan Russell, and Mathieu Aubry. Learning elementary structures for 3d shape generation and matching. In *Advances in Neural Information Processing Systems*, pages 7433–7443, 2019.

[142] Jan Bednarik, Shaifali Parashar, Erhan Gundogdu, Mathieu Salzmann, and Pascal Fua. Shape reconstruction by learning differentiable surface representations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4716–4725, 2020.

[143] Michael Kazhdan and Hugues Hoppe. Screened poisson surface reconstruction. *ACM Transactions on Graphics (ToG)*, 32(3):1–13, 2013.

[144] Nikhila Ravi, Jeremy Reizenstein, David Novotny, Taylor Gordon, Wan-Yen Lo, Justin Johnson, and Georgia Gkioxari. Pytorch3d. `https://github.com/facebookresearch/pytorch3d`, 2020.

[145] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[146] Yu Shen, Junbang Liang, and Ming C Lin. Gan-based garment generation using sewing pattern images. In *Proceedings of the European Conference on Computer Vision (ECCV)*, volume 1, page 3, 2020.

[147] Gül Varol, Javier Romero, Xavier Martin, Naureen Mahmood, Michael J. Black, Ivan Laptev, and Cordelia Schmid. Learning from synthetic humans. In *CVPR*, 2017.

[148] Greg Zaal. Hdri haven.

[149] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[150] Jean Feydy, Thibault Séjourné, François-Xavier Vialard, Shun-Ichi Amari, Alain Trouvé, and Gabriel Peyré. Interpolating between optimal transport and mmd using sinkhorn divergences. *arXiv preprint arXiv:1810.08278*, 2018.

[151] Justus Thies, Michael Zollhöfer, and Matthias Nießner. Deferred neural rendering: Image synthesis using neural textures. *ACM Transactions on Graphics (TOG)*, 38(4):1–12, 2019.

[152] Xiang Ni, Laxmikant V Kale, and Rasmus Tamstorf. Scalable asynchronous contact mechanics using charm++. In *Parallel and Distributed Processing Symposium (IPDPS), 2015 IEEE International*, pages 677–686. IEEE, 2015.

[153] Min Tang, Huamin Wang, Le Tang, Ruofeng Tong, and Dinesh Manocha. Cama: Contact-aware matrix assembly with unified collision handling for gpu-based cloth simulation. In *Computer Graphics Forum*, volume 35, pages 511–521. Wiley Online Library, 2016.

[154] Florence Zara, François Faure, and J-M Vincent. Parallel simulation of large dynamic system on a pc cluster: Application to cloth simulation. *International Journal of Computers and Applications*, 26(3):1–8, 2004.

[155] Pascal Volino, Nadia Magnenat-Thalmann, and Francois Faure. A simple approach to nonlinear tensile stiffness for accurate cloth simulation. *ACM Transactions on Graphics*, 28(4):Article–No, 2009.

[156] Cyril Zeller. Cloth simulation on the gpu. In *ACM SIGGRAPH 2005 Sketches*, page 39. ACM, 2005.

[157] David Baraff, Andrew Witkin, and Michael Kass. Untangling cloth. In *ACM Transactions on Graphics (TOG)*, volume 22, pages 862–870. ACM, 2003.

[158] Matthew Emmett and Michael L. Minion. Toward an Efficient Parallel in Time Method for Partial Differential Equations. *Communications in Applied Mathematics and Computational Science*, 7:105–132, 2012.

[159] Robert Speck, Daniel Ruprecht, Rolf Krause, Matthew Emmett, Michael L. Minion, Mathias Winkel, and Paul Gibbon. A massively space-time parallel N-body solver. In *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*, SC '12, pages 92:1–92:11, Los Alamitos, CA, USA, 2012. IEEE Computer Society Press.

[160] Daniel Ruprecht, Robert Speck, Matthew Emmett, Matthias Bolten, and Rolf Krause. Poster: Extreme-scale space-time parallelism. In *Proceedings of the 2013 Conference on High Performance Computing Networking, Storage and Analysis Companion*, SC '13 Companion, 2013.

[161] Martin J. Gander and Martin J. Gander. 50 years of time parallel time integration.

[162] Huamin Wang and Yin Yang. Descent methods for elastic body simulation on the gpu. *ACM Transactions on Graphics (TOG)*, 35(6):1–10, 2016.

[163] Marco Fratarcangeli, Valentina Tibaldo, and Fabio Pellacini. Vivace: A practical gauss-seidel method for stable soft body dynamics. *ACM Transactions on Graphics (TOG)*, 35(6):214, 2016.

[164] Bart Maerten, Dirk Roose, Achim Basermann, Jochen Fingberg, and Guy Lonsdale. Drama: A library for parallel dynamic load balancing of finite element applications. In *European Conference on Parallel Processing*, pages 313–316. Springer, 1999.

[165] Sergio Romero, Luis F Romero, and Emilio L Zapata. Fast cloth simulation with parallel computers. In *European Conference on Parallel Processing*, pages 491–499. Springer, 2000.

[166] Michael Keckeisen and Wolfgang Blochinger. Parallel implicit integration for cloth animations on distributed memory architectures. In *Proceedings of the 5th Eurographics conference on Parallel Graphics and Visualization*, pages 119–126. Eurographics Association, 2004.

[167] Bernhard Thomaszewski and Wolfgang Blochinger. Parallel simulation of cloth on distributed memory architectures. In *Proceedings of the 6th Eurographics conference on Parallel Graphics and Visualization*, pages 35–42. Eurographics Association, 2006.

[168] Florence Zara, François Faure, and Jean-Marc Vincent. Physical cloth simulation on a pc cluster. In *4h Eurographics Workshop on Parallel Graphics and Visualization*, 2002.

[169] Samantha Ainsley, Etienne Vouga, Eitan Grinspun, and Rasmus Tamstorf. Speculative parallel asynchronous contact mechanics. *ACM Trans. Graph.*, 31(6):151:1–151:8, November 2012.

[170] Rasmus Tamstorf, Toby Jones, and Stephen F McCormick. Smoothed aggregation multigrid for cloth simulation. *ACM Transactions on Graphics (TOG)*, 34(6):1–13, 2015.

[171] Miklós Bergou, Saurabh Mathur, Max Wardetzky, and Eitan Grinspun. TRACKS: Toward Directable Thin Shells. *ACM Transactions on Graphics (SIGGRAPH)*, 26(3):50:1–50:10, jul 2007.

[172] Matthias Müller and Nuttapong Chentanez. Wrinkle meshes. In *Proceedings of the 2010 ACM SIGGRAPH/Eurographics symposium on computer animation*, pages 85–92. Eurographics Association, 2010.

[173] Huamin Wang, Florian Hecht, Ravi Ramamoorthi, and James F O'Brien. Example-based wrinkle synthesis for clothing animation. In *ACM SIGGRAPH 2010 papers*, pages 1–8. 2010.

[174] Damien Rohmer, Tiberiu Popa, Marie-Paule Cani, Stefanie Hahmann, and Alla Sheffer. Animation wrinkling: augmenting coarse cloth simulations with realistic-looking wrinkles. In *ACM Transactions on Graphics (TOG)*, volume 29, page 157. ACM, 2010.

[175] Peter Schröder, Denis Zorin, T DeRose, DR Forsey, L Kobbelt, M Lounsbery, and J Peters. Subdivision for modeling and animation. *ACM SIGGRAPH Course Notes*, 12(2):43, 1998.

[176] Tony DeRose, Michael Kass, and Tien Truong. Subdivision surfaces in character animation. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pages 85–94. ACM, 1998.

[177] Charles Loop. Smooth subdivision surfaces based on triangles. 1987.

[178] Ladislav Kavan, Dan Gerszewski, Adam W Bargteil, and Peter-Pike Sloan. Physics-inspired upsampling for cloth simulation in games. In *ACM SIGGRAPH 2011 papers*, pages 1–10. 2011.

[179] Wei-Wen Feng, Yizhou Yu, and Byung-Uck Kim. A deformation transformer for real-time cloth animation. In *ACM Transactions on Graphics (TOG)*, volume 29, page 108. ACM, 2010.

[180] Igor Santesteban, Miguel A. Otaduy, and Dan Casas. Learning-Based Animation of Clothing for Virtual Try-On. *Computer Graphics Forum (Proc. Eurographics)*, 2019.

[181] Raquel Vidaurre, Igor Santesteban, Elena Garces, and Dan Casas. Fully Convolutional Graph Neural Networks for Parametric Virtual Try-On. *Computer Graphics Forum (Proc. SCA)*, 2020.

[182] Nicholas J Weidner, Kyle Piddington, David IW Levin, and Shinjiro Sueda. Eulerian-on-lagrangian cloth simulation. *ACM Transactions on Graphics (TOG)*, 37(4):1–11, 2018.

[183] Min Tang, Tongtong Wang, Zhongyuan Liu, Ruofeng Tong, and Dinesh Manocha. I-cloth: incremental collision handling for gpu-based interactive cloth simulation. *ACM Transactions on Graphics (TOG)*, 37(6):1–10, 2018.

[184] Junbang Liang and Ming C Lin. Time-domain parallelization for accelerating cloth simulation. In *Computer Graphics Forum*, volume 37, pages 21–34. Wiley Online Library, 2018.

[185] Tiantian Liu, Adam W Bargteil, James F O'Brien, and Ladislav Kavan. Fast simulation of mass-spring systems. *ACM Transactions on Graphics (TOG)*, 32(6):1–7, 2013.

[186] Doyub Kim, Woojong Koh, Rahul Narain, Kayvon Fatahalian, Adrien Treuille, and James F O'Brien. Near-exhaustive precomputation of secondary cloth effects. *ACM Transactions on Graphics (TOG)*, 32(4):1–8, 2013.

[187] Fabian Hahn, Bernhard Thomaszewski, Stelian Coros, Robert W Sumner, Forrester Cole, Mark Meyer, Tony DeRose, and Markus Gross. Subspace clothing simulation using adaptive bases. *ACM Transactions on Graphics (TOG)*, 33(4):1–9, 2014.

[188] Daniel Holden, Bang Chi Duong, Sayantan Datta, and Derek Nowrouzezahrai. Subspace neural physics: Fast data-driven interactive simulation. In *Proceedings of the 18th annual ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 1–12, 2019.

[189] Russell Gillette, Craig Peters, Nicholas Vining, Essex Edwards, and Alla Sheffer. Real-time dynamic wrinkling of coarse animated cloth. In *Proceedings of the 14th ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 17–26, 2015.

[190] Meng Zhang, Tuanfeng Wang, Duygu Ceylan, and Niloy J Mitra. Deep detail enhancement for any garment. *arXiv e-prints*, pages arXiv–2008, 2020.

[191] Victor J Milenkovic and Harald Schmidl. Optimization-based animation. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 37–46, 2001.

[192] Qianli Ma, Jinlong Yang, Anurag Ranjan, Sergi Pujades, Gerard Pons-Moll, Siyu Tang, and Michael J Black. Learning to dress 3d people in generative clothing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6469–6478, 2020.

[193] Jinlong Yang, Jean-Sébastien Franco, Franck Hétroy-Wheeler, and Stefanie Wuhrer. Analyzing clothing layer deformation statistics of 3d human motions. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 237–253, 2018.

[194] Tuanfeng Y Wang, Tianjia Shao, Kai Fu, and Niloy J Mitra. Learning an intrinsic garment space for interactive authoring of garment animation. *ACM Transactions on Graphics (TOG)*, 38(6):1–12, 2019.

[195] Edilson De Aguiar, Leonid Sigal, Adrien Treuille, and Jessica K Hodgins. Stable spaces for real-time clothing. *ACM Transactions on Graphics (TOG)*, 29(4):1–9, 2010.

[196] Bharat Lal Bhatnagar, Garvita Tiwari, Christian Theobalt, and Gerard Pons-Moll. Multi-garment net: Learning to dress 3d people from images. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 5420–5430, 2019.

[197] Alexandros Neophytou and Adrian Hilton. A layered model of human body and garment deformation. In *2014 2nd International Conference on 3D Vision*, volume 1, pages 171–178. IEEE, 2014.

[198] Marc Habermann, Weipeng Xu, Michael Zollhoefer, Gerard Pons-Moll, and Christian Theobalt. Livecap: Real-time human performance capture from monocular video. *ACM Transactions on Graphics (TOG)*, 38(2):1–17, 2019.

[199] Tao Yu, Zerong Zheng, Kaiwen Guo, Jianhui Zhao, Qionghai Dai, Hao Li, Gerard Pons-Moll, and Yebin Liu. Doublefusion: Real-time capture of human performances with inner body shapes from a single depth sensor. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7287–7296, 2018.

[200] Adult obesity facts. `https://www.cdc.gov/obesity/data/adult.html`, 2020.

[201] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

[202] Hao Zhang, Oliver Van Kaick, and Ramsay Dyer. Spectral mesh processing. In *Computer graphics forum*, volume 29, pages 1865–1894. Wiley Online Library, 2010.

[203] Theodore HH Pian and Pin Tong. Finite element methods in continuum mechanics. In *Advances in applied mechanics*, volume 12, pages 1–58. Elsevier, 1972.

[204] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.